Automatic and Accurate Detection of Webshell Based on Convolutional Neural Network

Zhuo-Hang Lv[1], Han-Bing Yan[1(✉)], and Rui Mei[2]

[1] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China
{lvzhuohang, yhb}@cert.org.cn
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
meirui@iie.ac.cn

Abstract. The rapid development of the Internet has changed the way people live and work. Web security, as the foundation of network security, has received much more attention. Based on the variability of Webshells and the vulnerability of detection methods, this paper proposed a model that used deep learning to detect and implements the automatic identification of Webshells. For the shortcomings of the traditional detection models using machine learning algorithms, this paper proposed to apply convolutional neural network to Webshell detection process. The deep learning model does not require complicated artificial feature engineering, and the modeled features trained through model learning can also allow the attacker to avoid targeted bypassing in Webshell detection. The experimental results showed that this method not only has better detection accuracy, but also can effectively avoid the attacker's targeted bypassing. At the same time, with the accumulation of training samples, the detection accuracies of the detection model in different application environments will gradually improvements, which has clear advantages over traditional machine learning algorithms.

Keywords: Webshell · Convolutional neural network · Text classification

1 Introduction

When an attacker conducts attacks such as penetration tests, data theft, dark chain implantation, and intranet lateral movement on the websites, the backdoors (that is, Webshells) of the website are often implanted on the website servers to maintain the management authority of the websites. Even if the website vulnerabilities are patched, as long as the backdoors of the hackers are exist, the hackers can still easily penetrate the website servers. There are many kinds of Webshells, small one that can exploit vulnerabilities, and big one that can obtain administrator privileges. Using a variety of attack tools and Webshell scripts, hackers can quickly and effectively implement bulk website intrusion. In addition, Webshell connection tools have different application

This work was supported by the NSFC Foundation, No. U1736218.

The original version of this chapter was revised: The acknowledgment of the NSFC Foundation was missing. The chapter has been updated to include these corrections. The correction to this chapter is available at https://doi.org/10.1007/978-981-13-6621-5_15

X. Yun et al. (Eds.): CNCERT 2018, CCIS 970, pp. 73–85, 2019.
https://doi.org/10.1007/978-981-13-6621-5_6

environments, such as “China Chopper”, “axe” and other tools are website management tools, and they are often used for website attack.

At present, the methods of Webshell detection are mainly divided into four categories. One is based on the experience of webmasters for manual identification, the second one is static feature detection, the third one is dynamic feature detection, and the last one is statistical analysis.

1.1 Manual Identification

Webmasters need to have a comprehensive grasp of the website pages and files, and have a high recognition ability for some newly added exception files, such as some special naming files, 1.asp, hello.php, abc. Jsp, etc. In addition, due to some common Webshell, such as “one sentence”, the file is very small, so attention should be paid to extremely small files. Finally, the content of the file is analyzed and determined. The normal webpage source files have a large number of labels and comments, and the layouts are neat and clear at a glance. The backdoor files, especially the small one, often have only some functions that perform specific functions, and the content is simple and the elements are very few.

1.2 Static Feature Detection

It is based on the features of the script files. These features generally include multi-dimensional information such as keywords, high-risk functions, file permissions, and owners. If the feature setting is reasonable, the success rate of this detection method will be high, but the disadvantage is that this method is only effective for the existing Webshells, and is basically undetectable for 0day Webshells. In such methods, machine learning algorithms are fully applied and are the mainstream of current Webshell detection. The application of machine learning algorithms needs to extract features from black and white samples (i.e. Webshell pages and normal web pages). The feature settings are based on the number of words, total length of text, key function calls, etc., and then apply different algorithms. Detection. For example, the literature [1] proposed a detection method using matrix decomposition, which has higher detection efficiency and correct rate, and can also detect new type of Webshells with a certain probability. However, the rationality and effectiveness of the classification method have not been confirmed when classifying page features. Literature [2] proposed a Webshell detection method based on decision tree, which can quickly and accurately detect the mutated Webshell, overcome the deficiencies of the traditional feature-based matching detection method, and combine the Boosting method to select the appropriate number of sub-models. The detection capability can be further improved. However, there are fewer training samples used in this document. In [3], a Webshell detection method based on Naive Bayesian theory is proposed for Webshell with obfuscated encryption coding technology. This model can accurately detect Webshells that have been confusingly encrypted and encoded, effectively improving traditional feature-based detection methods. The lack of detection methods is also the small number of training samples during the experiment, the training test samples need to be added, so that the classification model can more accurately identify the Webshell, and the classification model should be optimized and improved through experiments to improve the performance.

1.3 Dynamic Feature Detection

The dynamic detection method detects traffic requests, responses, system commands, and state changes generated in BS activities, discovers abnormal behaviors or states, and finally detects the existence of Webshell. For example, if there is a user accessing or calling a file that has never been used, the probability of a Webshell in the file is greatly increased. This method has certain detection capabilities for the new Webshells, but it is difficult to detect for some specific backdoors, and it is difficult to deploy. Intruders can also put Webshell into existing code, which makes the difficulty of dynamic detection more difficult. Literature [4] introduced a real-time dynamic detection method for PHP Webshell. For the key functions and variables involved in the execution of Webshell, mark tracking is performed by using a method similar to stain propagation to perform black and white discrimination.

1.4 Statistical Analysis

The statistics-based Webshell detection method is tailored to the user's access characteristics. The normal range of these features is statistically calculated and compared with the user-uploaded script files to finally determine the existence of Webshell. This method is still valid for encoded and encrypted Webshells, as these Webshells also exhibit some special statistical features. Generally, there are statistical analysis techniques such as coincidence index, information entropy, longest word length, and compression ratio. This method is generally used to identify obfuscated, encrypted code and performs well in identifying fuzzy codes or obscuring Trojans. However, there are also obvious shortcomings. Unblurred code is more transparent to statistical detection methods. If the code is integrated into other scripts, it is likely to be considered a normal file. Literature [5] proposed a Webshell detection technology based on semantic analysis. Compared with the rule-based detection method, the false positives are reduced, and the linear growth in time after the rule is increased is avoided. However, there is only one language in the literature is envolved. The scripting language was designed systematically, the system compatibility was not enough, and there were fewer training samples.

2 Convolutional Neural Network for Webshell Detection

2.1 Advantages of Convolutional Neural Networks

The advantages of CNN compared with other deep learning algorithms are as follows:

- Compared with RNN, its training time is shorter
- Compared with DNN, its parameters are fewer and the model is more concise.

The CNN model limits the number of parameters and mines the local structure. The training time is short and the effect is ideal. More importantly, compared with the traditional machine learning algorithm, the CNN model has the greatest advantage that its feature set is "learned" by itself. As long as the computing resources are sufficient, it

is not necessary to use statistical analysis data to find features. The advantages of applying CNN to Webshell detection are:

- As long as the sample quality is high, there can be a lower false positive rate.
- There is no clear feature extraction link, and the attacker could not bypass easily.
- Compared to traditional machine learning algorithms, CNN has better ability to discover 0day Webs hell or unknown attack scripts.
- The model is easier to accumulate and iterate. For new samples, just add them to training set.

2.2 Application in Text Processing

At first, the emergence of convolutional neural networks solved the problem that deep learning could not be done in the image processing field because the amount of computation was too large. The convolutional neural network greatly reduces the amount of computation of the network through convolution, weight sharing, pooling, etc., and the result is very satisfactory. The computer's storage of images is usually in the form of a two-dimensional array, and the convolutional neural network processes the small images by using a two-dimensional convolution function, so that advanced features can be extracted. Similarly, feature extraction and analysis of text segments can be performed using a one-dimensional convolution function, as shown in Fig. 1.

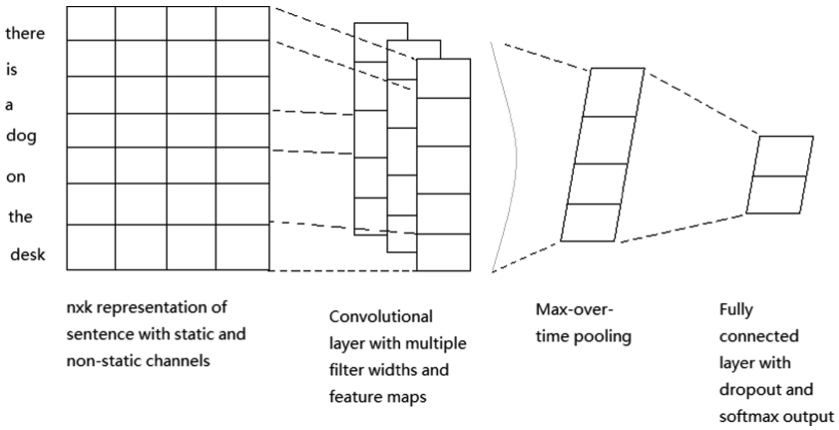

Fig. 1. Convolutional neural network for text processing

Assuming that $x_i \in R^k$ is a k-dimensional word vector corresponding to the i-th word in a sentence, a statement of length n can be expressed as:

$$x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n \tag{1}$$

Where $\oplus$ is the connector.

Thus, $x_{i:i+j}$ can be defined as a connection or combination of words or characters $x_i, x_{i+1}, \ldots, x_{i+j}$. Let $w \in R^{hk}$ be the filter in the convolution operation, also known as the convolution kernel, whose length is h, which can produce a feature after the convolution operation. For example, the feature c_i is generated by a word or character $x_{i:i+h-1}$ in a window.

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{2}$$

Where $b \in R$ is the offset term and f is a nonlinear function, such as a hyperbolic tangent function.

The convolution kernel is slidably convolved with the sentence $\{x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}\}$ to generate a feature layer.

$$c = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

Where $c \in R^{n-h+1}$, Then use the max-over-time pooling operation, that is, for each value in this pooled operation window, only the maximum value is reserved:

$$\widehat{c} = \max\{c\} \tag{4}$$

By such a method, only the most important features in the feature layer can be retained, thereby obtaining a pooled layer, and such a pooling operation can correspond to a statement with a variable length.

For the above steps, a convolution kernel can generate a feature after a convolution operation, and multiple convolution kernels can generate multiple features. The window sizes of these different convolution kernels can be different.

The pooled layer is then connected to the fully connected layer with dropout and softmax, and the final output is the probability distribution of each category [6].

2.3 Sample Data Preprocessing

One of the main application areas of machine learning related algorithms is text processing and analysis. However, the raw data form used for text cannot be directly used as input to the algorithms, because the original sample data is only a combination of characters, and most of the input of the algorithm cannot be a text file of different length, but a fixed-length vector. Therefore, the relevant text files need to be preprocessed, some of the most basic methods for extracting data numerical features from text content are:

- Mark the text content and encode the result of the tag using an integer value. In the process of marking, special characters or punctuation in the text can be used as the dividing point to split the text data.
- Count the frequency of occurrence of characters or marks in a text file.
- Add weights, for the marks that often appear in the sample file, reduce their weight, and the marks appearing in fewer samples increase their weight.

2.4 Simplified Word Segmentation

For Webshell detection, this paper first classifies the sample, treats the characters in the sample except English letters and Arabic numerals as separators, and then uses the bag-of-words model to encode the divided words, numbers, etc., to generate a dictionary, and then For each sample page, take the fixed number of character codes (such as 200) with the highest frequency appearing as the representative vector for this page, as shown in Fig. 2.

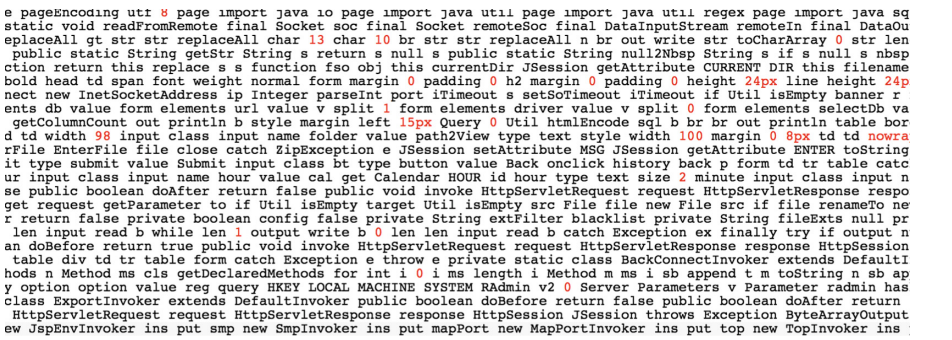

Fig. 2. Sample of word segmentation

2.5 Vectorization Model

One-Hot

The One-hot vector method first extracts the words in the sample set and extracts only the repeated words. This results in a vocabulary, assuming a size of V. The text is then represented by a vector of size V. If a word in the vocabulary appears in the text segment, the value in this dimension in the vector is 1, and no words appear in the text, the value of its corresponding bit is 0.

In the Webshell detection, this method is improved. Firstly, the dictionary is built. In order to avoid the sample matrix being too sparse, the dictionary size is controlled, and then the sample page is vectorized. Here the words that appear repeatedly in the text accumulate the corresponding bits on their vectors and words that do not appear in the simplified dictionary are ignored. This avoids excessive computational complexity and incorporates word frequency information.

Bag-of-Words

The so-called bag-of-words model is to treat the entire contents of a text file as a whole, and then add an index to all words, characters or positions in the whole. Thus, a text file can be represented by a word document matrix, where each column represents a word and each row represents a document. However, the disadvantage of this method of characterization is that:

- The matrix representing the document is too sparse in most cases and will consume a lot of storage resources.

- For the processing of a large number of different corpus samples, the representation of the document matrix will take up a lot of computing resources.
- The bag-of-words model ignores the relative positional information of words or characters in the text.

In view of the balance between the accuracy of the processing results and the computational complexity, this strategy can be optimized in special cases.

Word2vec

Word2vec is an NLP tool launched by Google in 2013. It is used to vectorize the words in the file, and the generated word vector can measure the relationship between the quantity and the distance, so word characterization and artificial word habits can be added to the process of vectorization.

In the past, neural networks were used to train word vector models. In order to calculate the classification probability of all words, such as the use of softmax in the output layer, you need to calculate the probability of softmax, and then find the maximum value. This process involves a very large amount of calculation.

For the Word2vec model, in order to avoid the heavy computation from the hidden layer to the output layer, the network structure has been modified and optimized. It uses the Huffman tree instead of the neuron structure in the output layer and the hidden layer [7]. In the Fuman tree, the number of leaf nodes is the size of the vocabulary composed of the input samples. At the same time, the leaf nodes have the same function as the neurons of the original output layer, and the internal nodes of the network act as the neurons of the original hidden layer. So, there is no need to calculate the softmax probability, which greatly reduces the amount of computation of the network.

Compared with the bag-of-words model, the Word2vec model incorporates the contextual relationship of lexical semantics, and the similarity between words can be obtained by calculating the Euclidean distance. This article uses the Word2vec library in Python. First, all the samples are trained to get the dictionary, and then each word in each sample is vectorized. In the process of vectorizing a single sample page, averaging and averaging all vectorized characters as a vector for this sample [8].

2.6 Convolutional Neural Network Structure

The convolutional neural network used in the experiments in this paper consists of the Embedding layer, the convolution layer, the pooling layer, the dropout layer, and the fully connected layer. The network is built on Tensorflow. TensorFlow is Google's second-generation artificial intelligence learning system based on DistBelief. It is most suitable for machine learning and deep neural network research, but the versatility of this system makes it widely used in other computing fields. The structure is shown in Table 1.

Table 1. Convolutional neural network structure

```
network = input_data(shape=[None,
MAX_DOCUMENT_LENGTH],name='input')
network = tflearn.embedding(network, input_dim=n_words+1, output_dim=128)
branch1 = conv_1d(network, 128, 14, padding='valid', activation='relu',
regularizer="L2")
branch2 = conv_1d(network, 128, 15, padding='valid', activation='relu',
regularizer="L2")
branch3 = conv_1d(network, 128, 16, padding='valid', activation='relu',
regularizer="L2")
network = merge([branch1, branch2, branch3], mode='concat', axis=1)
network = tf.expand_dims(network, 2)
network = global_max_pool(network)
network = dropout(network, 0.5)
network = fully_connected(network, 2, activation='softmax')
network = regression(network, optimizer='adam',
learning_rate=0.001,loss='categorical_crossentropy', name='target')
model = tflearn.DNN(network, tensorboard_verbose=0)
```

In the convolutional layer, setting padding does not add new elements based on the original data, that is, the boundary data is not processed, and the convolution is only performed in the original data.

The activation function uses ReLU:

$$f(x) = \max(0, x) \tag{5}$$

The advantage of using ReLU as an activation function is that its SGD will converge faster than tanh or sigmoid. ReLU can get the activation value based on only one threshold, no complicated operation, and it is linear. The disadvantage is that it is not suitable for inputs with large gradients during training, because as the parameters are updated, the ReLU neurons will no longer have an active function, which will cause their gradient to always be zero.

The regularization term uses the L2 norm, that is, each element in a vector is first summed to its square root, and then its square root is obtained. During the optimization process, the regularization term adds a penalty term to the activation value of the parameter in the layer, and the loss function together with this penalty term becomes the ultimate optimization goal of the network.

The pooling layer uses global_max_pool, that is, the feature point maximum pooling, and the maximum pooling can extract features better.

For the over-fitting problem in convolutional neural networks, the dropout layer is used to reduce its impact, which is equivalent to the effect of regularization. The essence of the dropout layer is to randomly delete some hidden neurons in the neural network. The input and output neurons are kept unchanged, and then the input data is forwardly propagated through the modified neural network, and then the error value is

propagated back through the modified neural network. However, after randomly deleting some hidden layer neurons, the fully connected network has a certain sparseness at this time, and finally the synergistic effects of different features are effectively reduced.

Classifier using softmax regression:

$$f(z_j) = \frac{e^{z_j}}{\sum_{i-1}^{n} e^{z_i}} \tag{6}$$

The dimension of the output vector is the number of required categories, and the value of each bit is the probability value of each one.

For the encrypted Webshells, such as the Base64-encoded Webshells, based on the above-mentioned bag-of-words model, has not been specially processed. After the word segmentation, the Base64-encoded part will be treated as a whole. The method does not reduce the final effect, and the same can be done for the other encoding encryption methods.

3 Experiments

3.1 Sample Collection

The so-called web page source code files are script files that can be parsed by the server side and written by the script language asp, jsp, php and so on. Common Webshells are also written by these scripting languages and then uploaded to the servers. The content of the webpage source file is shown as Fig. 3.

```
function sendpacket() //2x speed
{
  global $proxy, $host, $port, $packet, $html, $proxy_regex;
  $socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
  if ($socket < 0) {
    echo "socket_create() failed: reason: " . socket_strerror($socket) . "<br>";
  }
  else {
    $c = preg_match($proxy_regex,$proxy);
    if (!$c) {echo 'Not a valid proxy...';
    die;
    }
  echo "OK.<br>";
  echo "Attempting to connect to ".$host." on port ".$port."...<br>";
  if ($proxy=='') {
    $result = socket_connect($socket, $host, $port);
  }
  else {
    $parts =explode(':',$proxy);
    echo 'Connecting to '.$parts[0].':'.$parts[1].' proxy...<br>';
    $result = socket_connect($socket, $parts[0],$parts[1]);
  }
  if ($result < 0) {
    echo "socket_connect() failed.\r\nReason: (".$result.") " . socket_strerror($result) . "<br><br>";
  }
  else {
    echo "OK.<br><br>";
    $html= '';
    socket_write($socket, $packet, strlen($packet));
    echo "Reading response:<br>";
    while ($out= socket_read($socket, 2048)) {$html.=$out;}
    echo nl2br(htmlentities($html));
    echo "Closing socket...";
    socket_close($socket);
  }
  }
}
```

Fig. 3. Sample of source code

The Webshell sample in this article is mainly from related projects on Github, as shown in Table 2.

Table 2. Webshell related projects

Project name	Description
tennc/Webshell	This is a Webshell open source project
ysrc/Webshell-sample	Webshell sample
xl7dev/Webshell	Webshell & Backdoor Collection
tdifg/Webshell	Webshell Collection
testsecer/Webshell	A project for Webshell Collection

In addition, there are also common Webshell samples on the Internet, direct extraction from attacked websites, and samples shared by professionals. A total of three data sets of PHP, JSP, and ASP are collected:

- PHP Webshells: 2103
- JSP Webshells: 712
- ASP Webshells: 1129.

The white samples are derived from open source CMS, open source software, etc. Since there is no evidence that these open source software contain backdoor code, they are considered to be white samples. The collected data sets of PHP, JSP and ASP are as follows:

- PHP white samples: 3305
- JSP white samples: 3927
- ASP white samples: 3036.

3.2 Comparison of Three Vectorization Models

In this paper, the above three models are compared experimentally. In the processing of Webshell detection and classification tasks, the same structure of convolutional neural network is used. The final effect is shown in Table 3.

Table 3. Comparison of three vectorization models

Models	Accuracy
Improved One-hot (1000 dimensions)	90.43%
Improved One-hot (5000 dimensions)	97.26%
Improved One-hot (10000 dimensions)	98.39%
Bag-of-words (200 dimensions)	99.21%
Bag-of-words (400 dimensions)	99.31%
Bag-of-words (600 dimensions)	99.16%
Word2vec (average)	65.66%
Word2vec (sum)	70.23%

It shows that the improved one-hot vectorization model works well when the dictionary size is above 5000. The bag-of-words model is also very good, but the Word2vec model has the worst effect. It shows that the word2vec model is not suitable for document-based classification tasks. At the same time, the improved one-hot model consumes a longer time and consumes more computing resources when the dimension is very high. In contrast, the bag-of-words model is a simple and effective way to deal with it.

For the source code sample, since there is a difference in the writing language, the experiment uses a separate training method. First, for the PHP samples, the sample is trained firstly, and then the ten-fold cross-validation is used, as shown in Fig. 4.

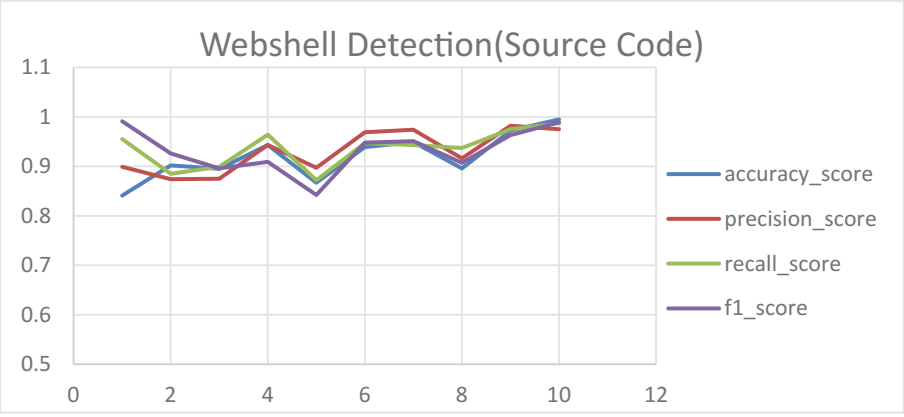

Fig. 4. Webshell source code detection curve

For the Webshell samples in various languages, the final indicators are shown in Table 4. The convolution function used in the experiment is 128 cores, one-dimensional, the processing length is 3, 4, 5 respectively, using relu as the excitation function, L2 norm processing over-fitting, and the dimension of the bag-of-words model is 400.

Table 4. Webshell sample testing indicators

Index	Accuracy	Precision	Recall	F1
PHP	99.5%	99.2%	99.7%	99.4%
ASP	98.3%	99.2%	99.5%	99.3%
JSP	97.5%	98.2%	99.4%	99.4%

It can be seen that the detection method based on convolutional neural network works pretty good in the application of Webshell detection. At the same time, due to different script languages, the generated lexicon is different. So Webshell source codes of different languages generate different detection models will have a better detection effect. The trained model is then compared with the existing detection methods.

The total number of Webshells used in this comparative experiment was 1,637, all written in PHP language. The detection accuracy of the convolutional neural network model compared with the decision tree, Webshell detector, D shield and 360 Trojan detection is shown in Table 5, in this case, the CNN network that has been used is the same as above:

Table 5. Comparison of test results

Methods	C4.5	D shield	360 detection	Webshell detector	CNN
Checkout	1347	1518	362	413	1553
Accuracy	82.28%	92.73%	22.11%	25.23%	94.87%

It shows that the trained CNN detection model has a higher detection accuracy.

3.3 The Impact of Filter Window

According to the research results of Zhang [9], for statements with a maximum word size of no more than 100, the size of the filter window in a convolutional neural network is generally between 1 and 10. But for statements with a maximum number of words over 100, the most appropriate window size (also known as a convolution kernel) will be larger. Moreover, for different data sets, there is a most suitable matching window size for each one. At the same time, the experiment confirms that more filtering windows with the same size that is near the most suitable size are added, the more final effect will be improved, but if the added filtering window sizes far apart from the most suitable size, the effect will be reduced. Based on this, for the 200-dimensional convolutional neural network model using the fixed sample vector of the bag-of-words model, the effects of different window sizes in the experiment are tried respectively. The results are shown in Table 6.

Table 6. Impact of filter window size on recall rate

Window size	5	9	12	15	30	60
Recall rate	82.4%	87.5%	85.8%	92.4%	83.2%	90.6%

The best window size for this experiment is 15.

Therefore, the window sizes of the convolution kernel in the experiment are 14, 15, 16. As shown in Table 7.

Table 7. Convolution kernel

```
branch1 = conv_1d(network, 128, 14, padding='valid', activation='relu', regularizer="L2")
branch2 = conv_1d(network, 128, 15, padding='valid', activation='relu', regularizer="L2")
branch3 = conv_1d(network, 128, 16, padding='valid', activation='relu', regularizer="L2")
```

4 Conclusion

This paper proposed the idea and process of using convolutional neural network model for Webshell detection. In this process, the most important thing is the quantity and quality of samples. A good training sample set can train very good models. Sample sets need to be expanded in the future. The training of the deep learning model does not require complex artificial feature engineering, which means that it is difficult for the attackers to bypass. Therefore, the deep learning model is stronger when facing some potential bypassing me thods. That is to say, the application of convolutional neural network to Webshell detection can prevent unknown attacks to a certain extent.

References

1. Dai, H., Li, J., Lu, X.-D., Sun, X.: Machine learning algorithm for intelligent detection of webshell. Chin. J. Netw. Inf. Secur. **3**(3), 71–77 (2017)
2. Hu, J., Xu, Z., Ma, D., Yang, J.: Research of webshell detection based on decision tree. J. Netw. New Media **6** (2012)
3. Hu, B.: Research on webshell detection method based on bayesian theory. Science Mosaic (2016)
4. Du, H., Fang, Y.: PHP webshell real-time dynamic detection. Netw. Secur. Technol. Appl. (2014)
5. Yi, N., Fang, Y., Huang, C., Liu, L.: Semantics-based webshell detection method research. J. Inf. Secur. Res. **3**(2), 145–150 (2017)
6. Kim, Y.: Convolutional neural networks for sentence classification. EprintArxiv (2014)
7. Goldberg, Y., Levy, O.: word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. EprintArxiv (2014)
8. Rong, X.: Word2vec parameter learning explained. Computer Science (2014)
9. Zhang, Y., Wallace, B.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. Computer Science (2015)