# Research on Network Personal Data Vulnerability Detection Technology Based on Deep Learning

Lei Ma[1(✉)] and Yunwei Li[2]

[1] Beijing Polytechnic, Beijing 100016, China
malei235@tom.com
[2] Beijing Youth Politics College, Beijing 100102, China

**Abstract.** Traditional data vulnerability detection technology has low detection accuracy in practical application. Therefore, this study designs a network personal data vulnerability detection technology based on deep learning. Firstly, the personal data set of network dynamic link library is established. In the data set, the vector characteristics of network personal data vulnerabilities are extracted by using deep learning network, and the vulnerability detection process is optimized. The test results show that this technology has high detection accuracy and low loss rate in the same data set. In the comparison of F value, this technology is also better than the traditional detection technology, which verifies that the network personal data vulnerability detection technology based on deep learning has high reliability in practical application.

**Keywords:** Deep learning · Network personal data · Data vulnerability detection · Vector features

## 1 Introduction

In the era of big data, the use of computers has greatly facilitated people's daily life and made people's communication more unimpeded. The utilization of online banking account, game account, credit card account and e-mail account has greatly improved people's quality of life, enabled people to enjoy the fun of life without leaving home, and the services of many network operators on the network have become richer [1]. But at the same time, it also brings opportunities for criminals.

As we all know, when registering an account, users need to fill in their own real information, which is often interrelated. It has great application value for both businesses and criminals, which enables some criminals to steal the user's account information through security holes, This will bring great security risks to the user's account security, resulting in the loss of user's account theft, credit card theft and so on.

The computer is the crystallization of the continuous development of human beings. The stability of the computer system is directly related to the user's experience of using the computer. However, it is precisely because the computer is invented by human beings


S.-H. Wang and Y.-D. Zhang (Eds.): ICMTEL 2022, LNICST 446, pp. 383–395, 2022.
https://doi.org/10.1007/978-3-031-18123-8_29

that the computer system will inevitably produce some security vulnerabilities, which will bring convenience to criminals, Many criminals will use the security vulnerabilities in the system to attack the user's computer, and then unknowingly obtain the user's personal information account accessed in the computer, which will undoubtedly pose a great threat to the user's personal information security [2]. At the same time, some criminals do not want to obtain illegal interests, and even attack the user's computer with the psychology of revenge, which will even implant viruses in the user's computer. These viruses will not only seriously affect the stability of the computer system, but also cause serious consequences such as computer system collapse and hardware burning.

Nowadays, in the era of big data, people no longer rely solely on computers to obtain data. The use of new intelligent terminal tools such as smart phones and industrial pads not only enriches people's daily life, but also lays a huge hidden danger for personal information security, Criminals or hackers often steal the user's personal information through security vulnerabilities in the intelligent terminal. For example, some software installation packages from unknown sources may contain various viruses. Once the user installs, these viruses will attack the intelligent terminal and steal the personal privacy stored in the intelligent terminal.

Against the above background, the relevant scholars have designed a web-based situational data mining leak detection technology, the technology by analyzing the data preparation, mining and evaluation shows three steps to do clustering processing data, provide help for vulnerability of data mining, the computation, and clustering results association mining method is applied to make tracking, clear scope of loopholes. Then design the overall structure framework of the technology, establish the detection database, and realize the design of the vulnerability detection technology from three aspects of static module, vulnerability scanning and page display module. Based on traditional research, this paper designs a network personal data vulnerability detection technology based on deep learning.

## 2 Detect Network Personal Data Vulnerabilities Based on Deep Learning

### 2.1 Establish Personal Data Set of Network Dynamic Link Library

Data set is a key and difficult point for personal data vulnerability mining based on deep learning. First of all, we often encounter the problem of unbalanced sample collection, because the number of vulnerable samples is much smaller than that of normal samples [3, 4]. Secondly, different types of data sets have different contributions to deep learning models. Data sets that perform well are often complex in the extraction process and expensive in time.

Aiming at some of the shortcomings of the current data set, this research proposes a data set with the granularity of assembly basic blocks and gives an idea of extraction. Aiming at the problem that there may be a large number of loops in the code block sequence, a descending loop method based on basic block clustering is proposed. Finally, a random oversampling method is proposed to alleviate the problem of unbalanced sample collection.

In the research of vulnerability mining based on machine learning for binary programs, the granularity of data sets used by predecessors includes program level, function call sequence level and function level. Different data sets will also have different effects on the performance of vulnerability mining models. The fine-grained data set is more conducive to the identification of the location of the vulnerability, and it also increases the difficulty of calculation and processing. In the process of establishing the dynamic data set of the personal link library, the format of each feature is set to "function name: parameter index = the parameter status at runtime" [5, 6]. First, store the declaration information of a large number of API functions locally, including information such as function names, parameter types, and function return value types. When a function breakpoint is captured in the process of tracing the program, the state type is subdivided according to the type of each parameter. If it is a pointer parameter, it is subdivided into a more precise pointer type through memory mapping information, such as Heap pointer, stack pointer, global pointer, etc.; if it is a data type parameter, the parameter $x$ in the data set is represented by $Num32B_n$, and the value that meets the following conditions is required:

$$2^n \leq x < 2^{n+1} \tag{1}$$

After calculating the value of $n$ in Eq. (1), it can be used as the basis and basis of vulnerability detection according to the actual situation.

## 2.2 Extract Vector Features of Network Personal Data Vulnerabilities

After feature vectorization, for a binary file, each word in the underlying language instruction becomes an $N$ dimensional vector, and this binary file is composed of many $N$ dimensional vectors [7–9]. However, the number of words contained in binary files of

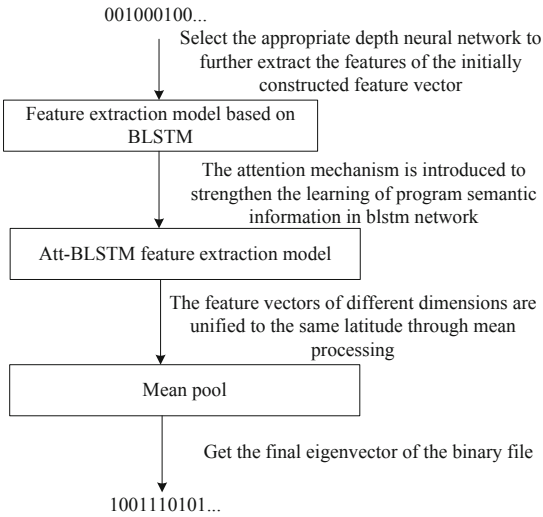


**Fig. 1.** Binary feature vector extraction process

different sizes is not the same, so the dimensionality of the feature vector representing each binary file is also different. Subsequent machine learning algorithms for vulnerability classification require that the feature vector of each sample has the same dimension, so this article divides the underlying language of the binary file according to the integer multiple of the sentence, and finally uses a mean pool to divide the characteristics of each binary file The vectors are unified into a fixed-size dimension.

The main solution to the problem is shown in Fig. 1.

Through the research on the construction of binary feature vector, this paper extracts the program information containing the context of words (in the underlying language instructions) from the binary file. Next, this paper selects the appropriate deep neural network to further extract the program information based on the context relationship between sentences (underlying language instructions).

With the segmentation of the underlying language, a binary file can be thought of as a text composed of serialized words. The process of extracting semantic information from such an article can be regarded as a natural language processing process. Therefore, the neural network suitable for natural language processing is also suitable for the vulnerability detection system in this paper. According to the characteristics of low-level language serialization, this paper first thinks of a cyclic neural network RNN that can learn sequence information [10]. However, RNN can not solve the problem of long-term dependence and can not remember the information of long-term sequence.

The propagation of vulnerability related data stream in the program may be a long process, especially the assembly code after binary disassembly is much larger than the source code, so a neural network that can remember long-term sequence information is needed. Therefore, this paper selects a variant of RNN that solves the long-term dependence problem, the long-term and short-term memory network LSTM, for feature extraction.

However, there is still a problem when using LSTM network. LSTM is a one-way network. It can well learn the front to back information in the sequence, but it can't learn

```
#include <stdio.h>
#include <ststring.h>

void vulfunc(char* str){
    char src[10];
    strcpy(src,str);
}
int main(){
    char* str = "AAAAAAAAAAAAAAAAAAAA"
    vulfunc(str);
    return;
}
```

**Fig. 2.** Stack overflow vulnerability

the back to front information. However, usually, the vulnerable part of a function may be affected by the previous program or the later program. Figure 2 shows a typical stack overflow vulnerability.

Figure 2 shows a typical UAF (Use After Free) vulnerability. The key to understanding this vulnerability is the system's memory management. After buf 1 is freed, the memory management will consider the address of buf 1 when it allocates buf 2 again. If the memory at the location is no longer useful, the memory will be allocated to buf 2, that is, the address of buf 2 points to the previous address of buf 1. The problem is that after the memory of buf 1 is released, no operation is performed to make the pointer of buf 1 NULL, which causes buf 1 to become a "wild pointer" and still point to the original memory address. In this way, if the pointer to buf 1 can be manipulated later, unexpected results can be produced.

The former is because there is no parameter check before calling the string copy function, and the latter is because there is no pointer nulling operation after calling the memory release function, which indicates that the forward and backward information of the program is related to the vulnerability, so It is not enough to use a one-way LSTM network. This article finally chooses to use a two-way LSTM network (BLSTM) for vulnerability feature extraction.

BLSTM is actually an extended structure of LSTM. Combining a forward LSTM and a backward LSTM network generates a BLSTM network. The BLSTM network can remember context information in both directions at the same time. Since the standard LSTM can only remember the above dependent information when processing the sequence, and the vulnerability may be related to the previous program or the subsequent program, it is necessary to add the following information of the program to the feature extraction training. BLSTM adds a reverse LSTM layer on the basis of the standard LSTM, and the information flow direction of the reverse LSTM layer is opposite to that of the standard LSTM layer. The schematic diagram of the BLSTM network is shown in Fig. 3.

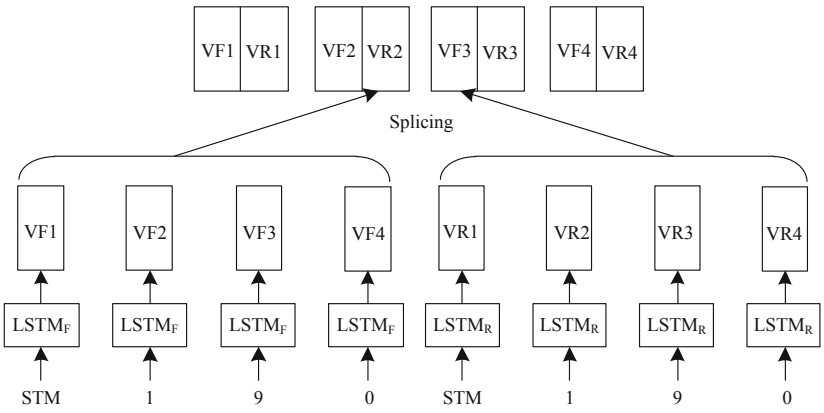


**Fig. 3.** BLSTM network principle diagram

Forward $\text{LSTM}_\text{F}$ input "STM", "1", "9", "0" in turn to get four vectors $\{VF1, VF2, VF3, VF4\}$. Input "0", "9", "1", and "STM" in the following $\text{LSTM}_\text{R}$ to get four vectors $\{VR1, VR2, VR3, VR4\}$. Finally, concatenate the forward and backward vectors to get:

$$\{[VF1, VR1], [VF2, VR2], [VF3, VR3], [VF4, VR4]\} \tag{2}$$

Which is:

$$\{v_0, v_1, v_2, v_3\} \tag{3}$$

In this paper, the training of the feature extraction model based on BLSTM is a supervised training. By training the BLSTM network to predict the next sequence of a long-term sequence in the underlying language to learn the context relationship between each assembly instruction of the underlying language, the purpose is to generate a representative The feature vector of each sample containing program semantic information. The feature extraction model diagram based on BLSTM is shown in Fig. 4.

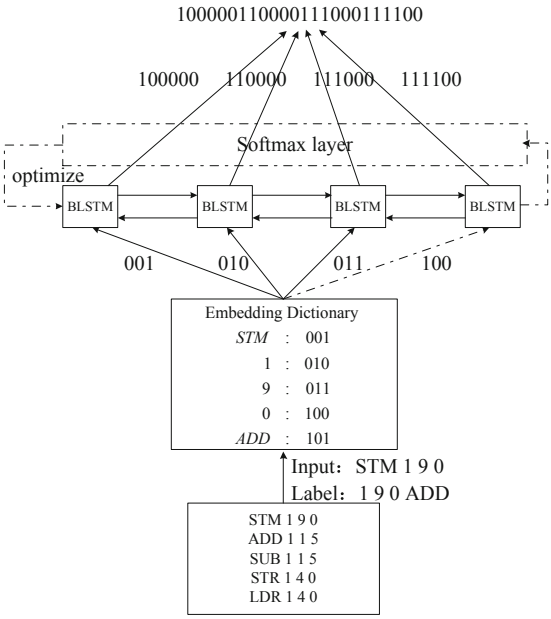


**Fig. 4.** The feature extraction model diagram based on BLSTM

Binary files of different sizes contain different numbers of words, so the dimensionality of the feature vector representing each binary file is also different. Some large binary files also contain a large number of words, which cannot be sent to the BLSTM network all at once. Learn. Therefore, this article first divides each binary sample. Each sentence of the underlying language contains 4 words, so this article divides each binary sample into pieces of data according to an integral multiple of 4, that is, in units of 4 × L words.

For each piece of data, 4 × L is used as the window size of the sliding window, and the data obtained by sliding back one grid is used as the label of this piece of data. Then according to the word embedding dictionary trained based on the feature quantization model of the underlying language, the word vector of each word in each data is searched in sequence order, and then sent to the BLSTM network layer for learning. The word vector corresponding to the label is sent to the Softmax layer, and the Softmax layer compares the predicted sequence with the label, calculates the deviation and optimizes the BLSTM network. Finally, the BLSTM network will output the state vector of each word, and splice the state vectors of these words together, which is the feature vector extracted from this piece of data that contains the context relationship between the underlying language instructions.

### 2.3 Optimize the Vulnerability Detection Process

The flow of the whole experiment is described from the perspective of the overall flow, as shown in Fig. 5.

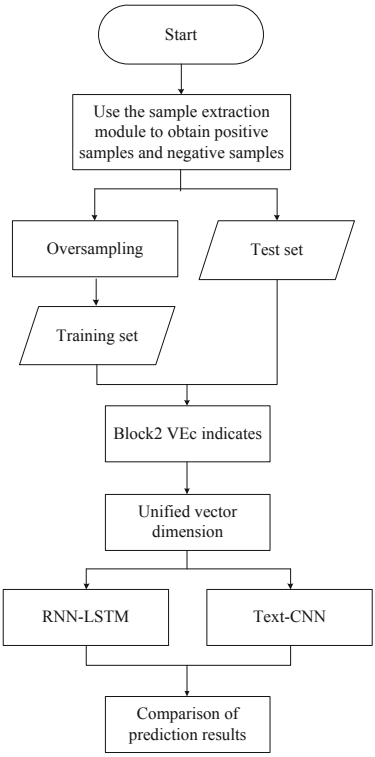


**Fig. 5.** Flow chart of vulnerability mining based on code block

The whole process involves the acquisition, preprocessing and training of data sets. The core idea is to convert the program into code block sequence, and then into recognizable word vector. In this way, vulnerability mining can be carried out through machine learning. The specific scheme is as follows:

(1) Collect normal programs and vulnerability programs, obtain the program execution process through the tracking program, and obtain the assembly code block sequence of descending cycle through static analysis as the characteristics of corresponding samples;
(2) The training set and test set are divided, and the improved random oversampling method is used to expand the vulnerability samples in the training set;
(3) Use Doc2Vec to learn the word vectors in assembly language, to B1ock2Vec representation of the assembly code blocks in the sample, and to unify the vector dimensions of the sample;
(4) The RNN-LSTM model and the Text-CNN model are used for training respectively, and the two models are tested on the test set, and evaluation indicators are given.

## 3 Experimental Test

In order to verify the practical application performance of network personal data vulnerability detection technology based on deep learning, the following experiments are designed.

### 3.1 Deep Learning Model Training

In the training process, this article uses Google's deep learning library TensorFlow and deep learning top-level library Keras as tools to build deep learning vulnerability detection models. Keras is an independently developed deep learning library, which encapsulates commonly used neural network layers, simplifies the model building and training process, and lowers the threshold of deep learning. Keras is a high-level neural network API, written in pure Python and based on Tensorflow, Theano and CNTK backends. Keras was born to support speedy experiments, simple and fast prototyping. Keras has the characteristics of high modularity, minimalism and scalability, and supports CNN and RNN or a combination of the two. Due to the excellent performance of Keras, this study chooses Keras to construct and train the deep learning model.

Since vulnerability detection is a binary classification problem, this article uses binomial cross loss as the objective function:

$$L(Y, P(Y|X)) = -logP(Y|X) \tag{4}$$

In addition, this article uses an efficient Adam optimizer to optimize the model. In each training step, 32 small batches of sampled data are randomly selected from the data set to update the model, so that the training process can be stabilized. In addition, this article uses the glorot uniform function to initialize the weights of the convolutional layer and LSTM layer.

The experimental training process runs on a computer with 4G Hz Intel i5-8250U CPU and 8 GB memory. After each vulnerability detection model is constructed, in order to better analyze the performance of the model, this paper compares the detection performance of the traditional multi-layer perceptron (MLP) model. The multi-layer perceptron (MLP) model used in this study has three hidden layers, and the number of neurons in each hidden layer is 1024, 512, and 128, respectively. The number of specific parameters and output dimensions of each network layer of the multilayer perceptron model are shown in Table 1.

**Table 1.** The structure and parameters of the MLP model

| Network layer | Output size | Number of parameters |
|---|---|---|
| Fully connected layer 1 | (None, 1024) | 208245 |
| Discard layer 1 | (None, 1024) | 0 |
| Fully connected layer 2 | (None, 512) | 548020 |
| Discard layer 2 | (None, 512) | 0 |
| Fully connected layer 3 | (None, 128) | 66645 |
| Discard layer 3 | (None, 128) | 0 |
| Fully connected layer 4 | (None, 1) | 192 |
| Active layer | (None, 1) | 0 |

The prediction accuracy curves in the training process of different models are shown in Fig. 6. It includes four deep learning models and a traditional multi-layer perceptron model.

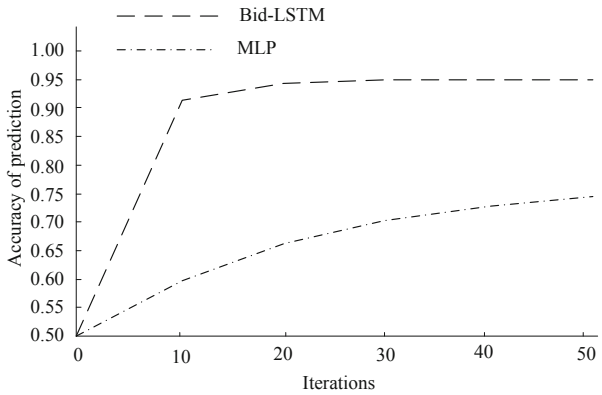


**Fig. 6.** The prediction accuracy curves of different models during the training process

In the first iteration, the accuracy curve of the deep learning model rises very fast. After training once on the training set, the prediction accuracy can reach more than 90%.

In the later iteration, the accuracy curve rises slowly and remains basically unchanged after the second iteration. Finally, the average prediction accuracy of the four deep learning models in this paper is about 90%.

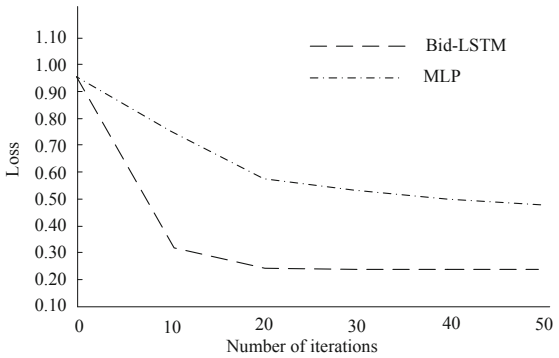


**Fig. 7.** Loss value curves of different models during training

Figure 7 shows the potential curves of loss function in the training process of different models. In this paper, the loss function bit of the deep learning model decreases significantly in the first iteration, reaching 0.2–0.3. In the following training process, the value curve of loss function decreases slowly and converges gradually, and finally stabilizes at about 0.18.

### 3.2 Test Results and Analysis

In order to avoid the uniformity of experimental results, the performance of the proposed method is compared with that of the traditional data vulnerability detection method based on network situation mining (Traditional method). The prediction accuracy and loss function values of the training set, verification set and test set in the whole training and testing stage are shown in Table 2.

**Table 2.** Experimental results of different detection technologies

| Data set | Technology of this article | Traditional technology |
|---|---|---|
| Training set accuracy | 1.0000 | 0.8261 |
| Validation set accuracy | 0.9865 | 0.746 |
| Test set accuracy | 0.9722 | 0.7366 |
| Training set loss | 0.2865 | 0.61 |
| Validation set loss | 0.4446 | 0.8349 |
| Test set loss | 0.5119 | 0.7068 |

According to the data in Table 2, after many iterations, the prediction accuracy of traditional detection technology finally reached 0.7366. Compared with the deep learning vulnerability detection technology proposed in this paper, the prediction accuracy is about 24% points lower, and the value of loss function is much higher than that of this method. According to the above experimental results, the detection accuracy of the traditional vulnerability detection technology in practical tasks is far lower than that of the technology in this paper. In conclusion, the application effect of the network personal data vulnerability detection technology based on deep learning designed in this paper is better.

In addition, the test data are divided into two categories: vulnerability (positive example) and no vulnerability (negative example), and the above vulnerability detection models are used to test the two types of test sets. The test results of the proposed technique and the traditional technique are shown in Table 3.

**Table 3.** Experimental results of different test data sets

| Detection technology | There are loopholes (positive example) | No loopholes (counter-example) |
|---|---|---|
| Technology of this article | 0.87 | 0.89 |
| Traditional technology | 0.72 | 0.59 |

According to Table 3, it can be seen that the detection results of the method in this paper are better than the traditional methods on the test sets with loopholes and without loopholes. In order to fully describe the performance of the model, it is not necessarily reliable to describe the detection accuracy and loss alone in the binary classification problem. Therefore, this paper calculates and compares the false positive rate (FPR, also known as false positive rate) of the model, in this paper Uniformly use "true and false, positive and negative" to express) and true rate (TPR, also called recall rate), and F-score (F-Score) as shown in Table 4, false positive rate (FPR) and true rate (TPR) It is a pair of elements that check and balance each other, and merge them into F-Score.

The specific calculation process of each result is as follows:

(1) True example (TP): The positive example predicted by the model to be positive, that is, the example whose label is positive and the predicted result is also positive.
(2) True Negative Example (TN): A negative example predicted by the model to be negative, that is, the label is negative and the predicted result is also negative.
(3) False positive example (FP): A negative example predicted by the model to be positive, that is, an example with a negative label and a positive prediction result.
(4) False Negative Example (FN): The positive example predicted to be negative by the model, that is, the example whose label is positive and the predicted result is negative.
(5) False positive rate (FPR): Solve by the above results, as follows:

$$FPR = FP/(TP + FP) \tag{5}$$

The number of negative sample results that are predicted to be positive accounts for the proportion of the number of results that are predicted to be positive. The lower the value, the better the performance of the model.

(6) True rate (TPR): The calculation result is as follows:

$$TPR = TP/(TP + FN) \tag{6}$$

The number of positive sample results that are predicted to be positive/the actual number of positive samples. The higher the value, the better the performance of the model.

(7) F-Score: As shown in formula 4.

$$F - Score = 2TP/(2TP + FP + FN) \tag{7}$$

Through the above calculation methods, the calculation results of different technologies are shown in Table 4.

**Table 4.** F-Score numerical calculation results

| Detection method | False positive rate(FPR) | True rate(TPR) | F value(F-Score) |
|---|---|---|---|
| Technology of this article | 0.1329 | 0.8956 | 0.8642 |
| Traditional technology | 0.4337 | 0.6551 | 0.635 |

As can be seen from Table 4, the false positive rate of the detection technology in this paper reaches 0.1329, and its true rate is also relatively high, reaching 0.8956. In the comparison of F values, the test results of the proposed technique are also better than the traditional technique, which highlights the application advantages of the proposed technique.

## 4 Conclusion

In this paper, a deep learning model is optimized for vulnerability detection tasks, and the network structure and hyperparameters of each layer of the model are designed. Through training on the collected 40000 data sets and finally testing on 5000 data sets, the deep learning vulnerability detection model can achieve high detection accuracy. Compared with the multi-layer perceptron model of traditional machine learning methods, the accuracy of this paper is greatly improved. By comparing the true rate, false positive rate and F value of different models on the test set, it can be seen that the F value of convolutional neural network model is the highest.

By analyzing the prediction accuracy curve and loss function value curve of the training process, it is found that the parameters of the deep learning model based on convolutional neural network are generally less, and the training speed is significantly faster than that based on long-term and short-term memory network. The comparison of the final performance of each model shows that the convolutional neural network model has the best comprehensive performance. To sum up, this paper has achieved effective results.

## References


1. Shen, G.: Vulnerability of vulnerability defense control simulation of network system resource data. Comput. Simul. **37**(04), 308–311 (2020)
2. Wang, J., Liu, J., Ma, Y., et al.: An automated detection and verification method for webview component vulnerabilities. Trans. Beijing Inst. Technol. **40**(02), 169–174 (2020)
3. Gong, K., Zhou, Y., Ding, L., et al.: Vulnerability detection using bidirectional long short-term memory networks. Comput. Sci. **47**(05), 295–300 (2020)
4. Shao, S., Wang, M., Chen, D., et al.: Analysis of android application component exposure vulnerability based on machine learning. Trans. Beijing Inst. Technol. **39**(09), 974–977 (2019)
5. Li, Y., Feng, D.: Laser sensor network malicious code active detection system design. Laser J. **40**(06), 212–215 (2019)
6. Qi, L.: Mobile network hybrid security vulnerability detection simulation under static defense. Comput. Simul. **36**(04), 282–285 (2019)
7. Chen, B., Li, H., Li, B.: Application research on pseudo measurement modeling and AUKF in FDIAs identification of distribution network. Power Syst. Technol. **43**(09), 3226–3236 (2019)
8. Xia, Z., Yi, P., Yang, T.: Static vulnerability detection based on neural network and code similarity. Comput. Eng. **45**(12), 141–146 (2019)
9. Deng, Z., Lu, Y., Huang, Z, et al.: Network program vulnerability detection technology based on program modeling. J. Beijing Univ. Aeronaut. Astronautics Astronau **45**(04), 796–803
10. Li, Y., Cui, Y. ,Lv, J., et al:.Combined deep learning method for open source software vulnerability detection. Comput. Eng. Appli. (11), 52–59 (2019)