Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 129 (2018) 248–256

Procedia Computer Science

www.elsevier.com/locate/procedia

2017 International Conference on Identification, Information and Knowledge in the Internet of Things

# ASSCA: API based Sequence and Statistics features Combined malware detection Architecture

Lu Xiaofeng[a,*], Zhou Xiao[a], Jiang Fangshuo[a], Yi Shengwei[b], Sha Jing[c]

[a]*School of Cyberspace Security, Beijing University of Post and Telecommunications, Beijing 100876;*
[b]*China Information Technology Security Evaluation Center, Beijing 100085;*
[c]*Key Lab of Information Network Security of Ministry of Public Security, Shanghai 201204;*

**Abstract**

Dynamic analysis of malware sample is an important method in the malware detection. In this paper, a malware detection architecture is proposed that combines machine learning and deep learning. The combination classification architecture focuses on the dynamic behavior of a malware sample. A new feature extraction method is proposed for dynamic behavior analysis. In this paper, recurrent neural network model is used to extracts the abstract features. Several sequence data preprocessing methods are studied to remove the redundant data. Experiments show that the AUC of the combination architecture is 99.3%. The classification performance of the combination architecture is better than the separate machine learning or deep learning.


Selection and peer-review under responsibility of the scientific committee of the 2017 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2017).

*Keywords:* malware Classification; Machine learning; deep learning; call sequence

## 1. Introduction

With the rapid development of the Internet, malicious software has grown rapidly in terms of categories and quantities, and the propagation modes have constantly updated. Detection unknown malware has become a new challenge. In recent years, there have been more and more APT attacks and senior virus for industrial control systems. A large number of variants of these viruses have made it difficult to detect unknown virus based on fixed features, and the information security of industrial control systems has become increasingly prominent.

* Lu Xiaofeng. Tel.: 18618264185;
*E-mail address:* luxf@bupt.edu.cn


Selection and peer-review under responsibility of the scientific committee of the 2017 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2017).
10.1016/j.procs.2018.03.072

With the development of artificial intelligence technology, researchers proposed to use machine-learning technology to detect and classify malicious sample [1].We combined the LSTM (Long Short Term Memory) deep learning model [2] and random forests model and proposed a malware detection architecture, ASSCA. The detection architecture employs both API sequence features and statistical features. The contributions of this paper are as follows:

（1） This paper proposes a system architecture that combines LSTM deep learning model based on sequence data and machine learning model based on API statistical features. The traditional machine learning model and the recurrent neural network model are integrated to obtain a better classifier.

（2） This paper proposes a new method to extract features from the API sequence and to optimize it.

（3） This paper studies the problem of long sequence in recurrent neural network and proposes several new methods of removing redundant data from API call sequence.

## 2. Related Work

The related work is divided into two parts: the malware detection method based on machine learning and the detection method based on deep learning

### *2.1. Malicious detection based on machine learning*

Researchers use different behavioral feature as input to their statistical models. Huang et al. [3] used dynamic and static methods to analyze and extract feature of the malicious sample. Then he use KNN(k-Nearest Neighbor), decision tree and SVM(support vector machine) method for classification. Liu et al. [4] mainly uses the static features obtained by anti-compilation APK. These static features include sensitive API, string, and application permissions and certificates. Neural network is used for detection. Yang et al. [5] proposed an improved random forest algorithm. Besides the static features, Zhang et al. [6] also used Opcodes. He used the N-gram model to deal with Opcodes and classified malwares together with the features such as permissions. Santos I et al. [7] mainly used the static feature of PE files. Ravi C et al. [8] made n-gram processing for Opcode sequences that are extracted from PE files, and used decision tree, KNN, Naive Bayes and SVM to classify malwares. Most of the above methods based on static features. For the research on dynamic analysis, the N-gram method is used to process the long sequence to obtain the eigenvector. In this paper, a new API sequence processing method is proposed for malicious sample detection.

### *2.2. Malicious detection based on deep learning*

In recent years, malicious sample detection based on deep learning has become a hot topic. G. E. Dahl et al.[10] used the neural network method to classify malicious samples with large-scale data sets. J. Saxe et al. [11] used the forward neural network to classify the static analysis samples, but they did not take into account the dynamic analysis, namely API call sequence. After the forward neural network, the deep learning model of CNN (convolution neural network) and RNN (Recurrent Neural Network) with their improved versions became the focus of malicious sample detection. Many studies used CNN to extract dynamic API call sequence feature and RNN to classify malwares [12]. Microsoft researcher Jack W Stokest et al. [14] proposed to use both the operating system API and C library API. In the above literatures, the length of the API call sequence are very different, the length can reach 1 to 1000000.The automatic algorithms that can preserve the call sequence features and reduce the sequence length are less studied. The literature [14] directly truncated the call sequence data to make the sequence data be constant length. Only two API calls are retained to continuous same calls in [13]. In this paper, the automated algorithm of reducing sequence length is studied.

## 3. Combination Classification Architecture

### *3.1. System Description*

The API based Sequence and Statistics features Combined malware detection Architecture (ASSCA) is as shown in figure1. A sandbox executes a malware to get the API call sequence. The architecture includes a machine learning

model which is the random forest model and a deep learning model which is long and short term memory model (LSTM). The random forest model is used classification the system call sequences after the association analysis. The system call sequence is preprocessed to obtain the numerical feature vector, then the feature vector inputted into the LSTM model for classification. Tensorflow framework is used train and test the LSTM model [15].

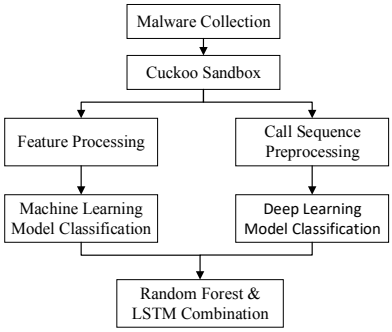



Figure 1. Malware Detection Architecture

### *3.2. Deep Learning Model Classification*

We utilize the LSTM model to extract the abstract features and logistic classifier to classify malicious samples.

#### *3.2.1. Sequence data preprocessing*

Each API function represented by a specific integer [3] for the subsequent data processing .

##### *3.2.1.1. Redundant subsequence removal method*

(1) N-gram sample subsequence extraction

By analyzing of the malware API sequences, we find there are a large number of subsequences in the sequence that have little effect on the classification malware. Removal these subsequences can make the classification faster and more accurate. In this paper, the text-based N-gram method is used extract the subsequences. The basic idea of N-gram algorithm is to carry out the sliding window operation with the size of N in the API sequence stream. table 1 is a sequence of API functions and the corresponding integer sequence. The text-based N-gram method extracts the sub-sequences, where N is 4, the sliding window moves with step being 1.

Table 1.4-Gram subsequence extraction

| Index | API function | 4-Gram |
|---|---|---|
| FindResourceExW | 24 | 24 26 18 18 |
| NtProtectVirtualMemory | 26 | 26 18 18 6 |
| NtProtectVirtualMemory | 18 | 18 18 6 13 |

(2) Remove redundant subsequences by information gain

The dimension of subsequences list extracted using the N-gram method is very large. Only those subsequences that have evident effect on the classification malware selected. In this paper, the feature selection method based on information gain utilizes to select the key subsequences.

Information entropy is a measure of system information. For a feature, system information entropy will change when the system know it and ignore it, the difference between information entropy is the information gain that this feature brings the system. The C entropy of the malware detection system defined as:

$$H(C) = -\sum_{i=1}^{n} p(c_i) \log p(c_i) \quad (1)$$

where $p(c_i)$ is the proportion of category $c_i$ samples.

The information gain of the subsequence T to class C is:

$$IG(T) = H(C) - H(C|T) = H(C) - p(t_i)H(C|t_i) - p(t_j)H(C|t_j) \quad (2)$$

$H(C)$ is the information entropy of category C, $H(C/t_i)$ is the information entropy of the sample classification under the condition of the feature T appears, $H(C/t_j)$ is the information entropy of the sample classification under the condition of the feature T doesn't emerge, $p(t_i)$ is the probability that the feature T appear. $p(t_j)$ is the probability that the feature T doesn't emerge.

(3) Valid subsequence connection

The feature subsequences selected by the information gain are called valid subsequences. The valid subsequences collection is called a dictionary. The following table 2 shows a 4-gram subsequence connection example. Extract the 4-gram subsequences that are in the dictionary from all the subsequences, and then connect them in chronological order to get the API call sequence of the sample. The API call sequence removes the subsequence with little effect on the classification. This method greatly reduces the length of the system call sequence of a sample. As the sequence length for the LSTM model is reduced, the LSTM model training and detection are faster and more accurate.

Table 2. 4-Gram subsequence connection

| 4-Gram subsequence | 4-Gram subsequence | Connection |
|---|---|---|
| $S_1S_2S_3S_4$ | $S_2S_3S_4S_5$ | $S_1S_2S_3S_4S_5$ |
| $S_1S_2S_2S_2$ | $S_2S_2S_2S_2$ | $S_1S_2S_2S_2S_2$ |

*3.2.1.2. Continuous same API pattern removal*

In order to reduce the length of the numerical sequence entered into the LSTM model, without information entropy on the system call being not reduced, [13] used proposed to remove the continuously same API. We find there still is information redundancy after this operation. For example, in the API sequence of S1S2 S1S2 S1S2, 'S1S2 'means an API pattern. There are lots of continuously same pattern of API call. As shown in table 3 below, we take the method of removing the continuously same pattern of API call.

Table 3. continuous same mode API removal

| continuous same mode API | continuous same mode API removal |
|---|---|
| $S_1S_2$ $S_1S_2$ $S_1S_2$ | $S_1S_2$ |
| $S_1S_2S_4S_3$ $S_1S_2S_4S_3$ | $S_1S_2S_4S_3$ |

*3.2.2. Deep learning model*

In the RNN model, each hidden layer output retained in the network, and the output at the next moment is determined together with current output and the input at next moment. This recycling makes the RNN great success in processing time sequences such as machine translation. The LSTM model is an improved version of the RNN model. It overcomes the back propagation gradient diffusion or gradient explosion. It is more suitable for dealing with long time sequences data. The LSTM model is used the experiment. The deep learning model structure shown in figure 2 below.

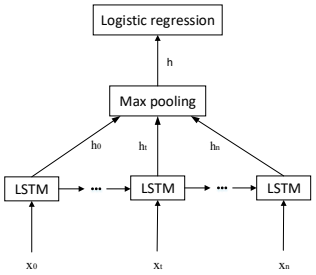



Figure 2. LSTM classification model structure

*3.2.2.1. LSTM model*

The input in LSTM model is time sequence X={$x_0$,$x_t$, ... $x_n$},where $x_t$ is the input at a moment, which is a system call function in a sample obtained by preprocessing. It is represented by the one-hot vector. $W_i$, $W_f$, $W_c$, $W_o$, $U_i$, $U_f$, $U_c$, $U_o$, $V_o$is the weight matrix, $b_i$, $b_f$, $b_c$, $b_o$ is the offset vector.

Firstly, calculate the input gate, memory gate of the middle state value, forget gate at moment *t*.

$$\begin{cases} i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ C_t' = \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \end{cases} \quad (3)$$

Secondly, calculate the memory gate status values, as well as the output gate and hidden layer output

$$\begin{cases} C_t = i_t * C_t' + f_t * C_{t-1} \\ o_t = \sigma(x_t + U_o h_{t-1} + b_o) \\ h_t = o_t * \tanh(C_t) \end{cases} \quad (4)$$

The hidden layer $h_t$ contains the feature information of the call sequence, and the feature information is input into the classification model for malicious sample detection.

*3.2.2.2. Max-Pooling and Classification*

It is important to deal with the hidden layer feature information and to obtain abstract features that represent the sequence information. The Max-Pooling algorithm can extract hidden layer features. Max-Pooling is used in our study to get the eigenvectors with certain length for the classifier and to reduce the number of dimensions without losing the hidden layer information. Max-Pooling uses equation (5) to obtain the eigenvector.

$$h_{\max}(i)=\max(h_0(i), h_1(i),\ldots h_n(i)) \quad (5)$$

$i \in (0,1,2,\ldots N-1)$,N is the number of neurons in per hidden layer. Finally, the eigenvector input to the classification model is connected by the output of the Max-Pooling and the last state vector of the hidden layer is $[h_{max},h_n]$.

Finally, the feature vector obtained from Max-pooling input into the classifier. As the malicious samples are classified into two classification in this study, logistic regression is selected as the classifier.

*3.3. Machine Learning Model Classification*

For system calls, we need analyze the relationship between system calls. In this paper, we propose an algorithm to calculate the association between two API calls quickly based on parameter hash, AMHARA (ArguMent-Hashing based Api coRrelation fast Analysis algorithm). The algorithm for the current traverse API $s_i$, it reverse traverses T API Calls. In a reverse traversal call $s_j$($T < j < i$), the parameters of $s_i$ and $s_j$ are compared, as formula (6) shows. The function return 1 when the input parameters are same, otherwise return 0. The length of some parameters is large such as buffer. So we hash the parameters and directly compare hash values of parameters as formula (7) shows. After the algorithm executed, it gets a dataset of associated call pairs that not only records the associated call pairs, but also records the number of the associated call pairs.it can as the feature input the classification. Random forest is easy to carry out large-scale parallel training on distributed computing platform. So in the choice of classifiers, the random forest is used a classifier, which got a good performance in the experiment.

$$S_{i,j} = \sum_{k=0}^{m} \sum_{l=0}^{n} Compare(s_i^k, s_j^l) \quad (6)$$

$$S_{i,j} = \sum_{k=0}^{m} \sum_{l=0}^{n} Compare[hash(s_i^k), hash(s_j^l)] \quad (7)$$

## 4. Experiment

*4.1. Evaluation Indicators and Data*

In the binary classification algorithm, we use ROC, AUC and ACC as the evaluation indicators.

The experiment samples come from four data sets: normal samples from windows7 and windows xp system exe files, the malicious samples were randomly picked up from the Virus Share[16], VirusTotal [17]. The labels of samples

derived from VirusTotal.com after a weighted probability calculation to the detection results of VirusTotal.com.

### 4.2. Deep Learning Experiment

#### 4.2.1. Experimental parameters

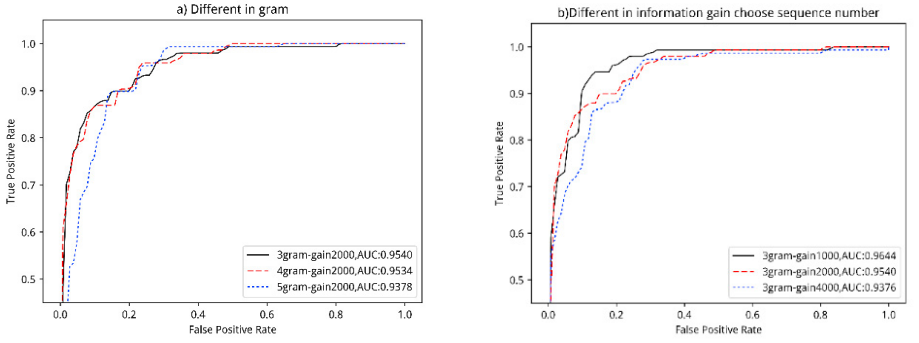
In the experiment, the samples are divided into training set, verification set and test set. The experiment parameters are divided into LSTM model parameters and Logistic classification parameters as shown in table 4.

Table 4. experiment parameter

| Indicator | LSTM model parameter | Logistic classification parameter |
|---|---|---|
| Input | Input dimension：200(500,1000 …)<br>dropout:0.5<br>embedding dimensions：100 | Input dimension：1000 |
| Middle layer | dropout:0.5,<br>memory dimension:500<br>forget gate bias：1.0 | —— |
| Others | Initial learning rate：0.001<br>Gradient clip：1 | w: mean 0, standard deviation 0.1<br>b:0.1 |

#### 4.2.2. Deep learning results

Different classification results obtained using different preprocessing methods in the experiment as shown in figure 3. Figure 3 (a) is the results obtained using different N-Gram methods with using the redundant subsequence removal method described in 3.2.1. It shows that the best result is with 3-Gram method, AUC is 0.9539. 3-gram method was used in [3] to obtain the best results, but it's accuracy rate is 85% -90%. Figure 3 (b) is the result obtained by selecting subsequences of different numbers using the information gain method. It shows that the AUC is largest when the number of the selection subsequence is 1000 during the information gain operation. The smaller number of subsequence by the information gain indicates that the more classification information contained in these subsequences when the length of the system call sequences is fixed. Figure 3 (c) shows that call sequences length is cut to 200, 500 and 1000 respectively. It used 3-Gram and the number of subsequence by information gain is 1000. It seen the longer the sequence, the larger the AUC. However, the train time is longer for a long call sequence than a short call sequence. Figure 3 (d) shows the results of different preprocessing methods. The experiment shows that the results of both preprocessing methods are better than without preprocessing. The AUC of the preprocessing of redundant subsequence removal is largest.



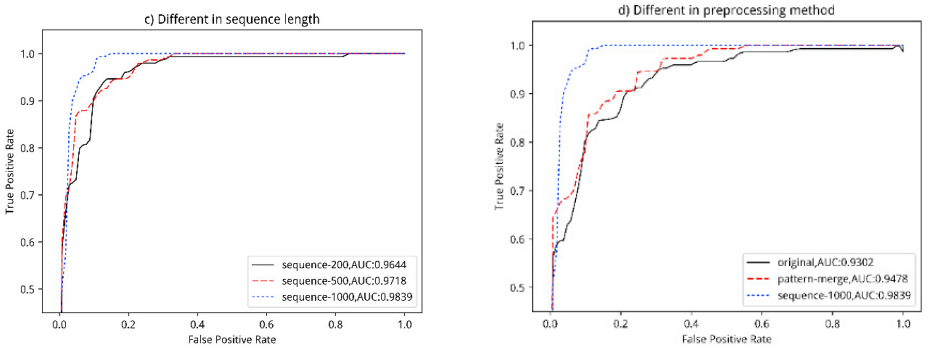



Figure 3. LSTM experiment

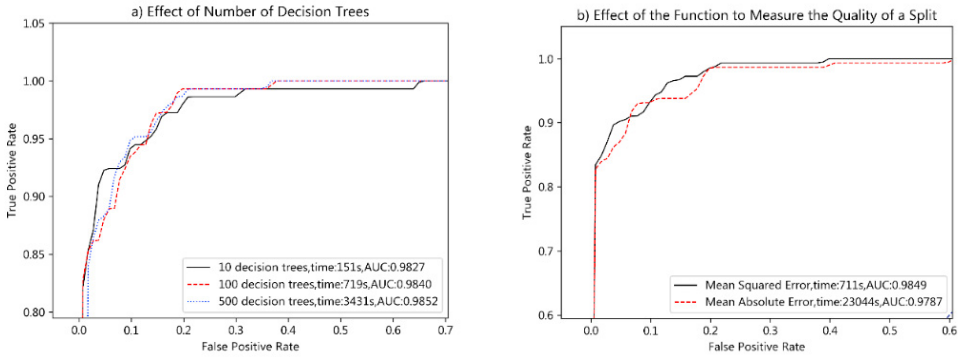
### *4.3. Random Forest Experiment*

#### *4.3.1. Experimental parameters*

The parameters used by the algorithm mainly have the number of decision trees in the random forest, the decision tree split evaluation methods, Bootstrap sampling parameters, the feature quantity parameter.

#### *4.3.2. Machine learning results*

First, we study how the number of decision trees effect the overall performance of the random forest. In the experiment, we found that the AUC value increased as the number of decision trees increasing as shown in figure 4(a). At the same time, the consumed time in the training process increases as well. For the decision split evaluation method, figure 4(b) shows that the result of the Mean Square Error (MSE) is better than the average absolute error (MAE). The reason is that the power operation of MSE reduces the influence of a small error in the feature calculation on the result. Figure 4(c) Bootstrap sampling in the decision tree construction processing impacts on the AUC values. After using the Bootstrap method, the classifier is more efficient. In order to study the effect of the feature quantity of the optimal segmentation, we compared the effects of different numbers of features on the experimental results. The experimental results shown in figure 4(d). According to the experimental results, the training consumed time is proportion to the number of features.



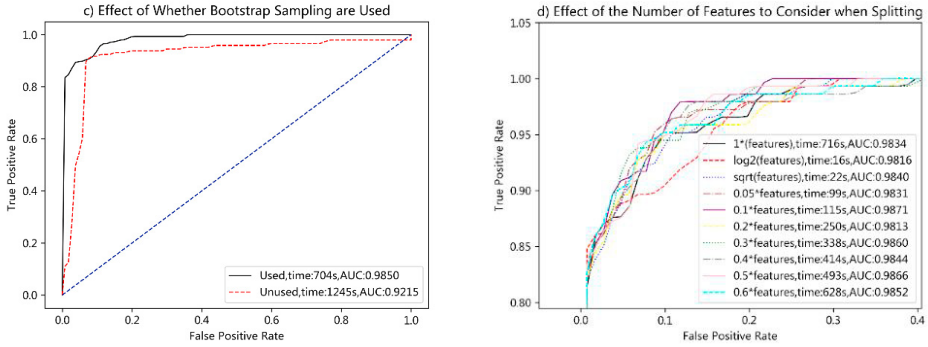



Figure 4 . random forest experiment

### *4.4. Random Forest and LSTM Combination*

By the random forest with parameter optimization, the highest AUC is 0.987. The AUC of the deep learning LSTM model was 0.983. We combined machine learning and deep learning. A sample is studied by machine learning model and deep learning model respectively, and then two pairs of probability values and prediction results are got. If the difference of a probability value to the threshold is larger than another probability value, it's prediction result is final prediction result. Figure 5 shows the AUC of the machine learning model and deep learning model combination is 0.993, which is better than the above separate experimental results.

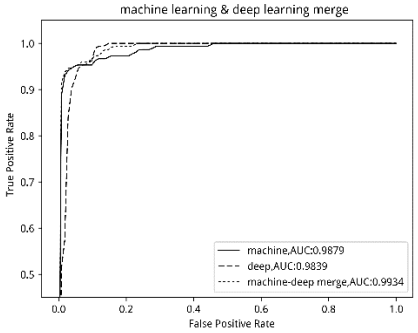



Figure 5. machine learning &deep learning

The following table 5 shows the accuracy of different algorithms (ACC). The ACC of the combining machine learning and deep learning in this paper is 0.957. The experiment shows that the classification performance of machine learning & deep learning combination architecture in this paper is better than others.

Table 5. The accuracy of different algorithms

| Algorithms | ACC |
|---|---|
| Decision tree [3] | 0.950 |
| Iterative Learning [8] | 0.900 |
| Neural Network [10] | 0.9047 |
| CNN + LSTM[13] | 0.894 |
| KNN [10] | 0.941 |
| Random Forest [6] | 0.953 |
| Machine learning & Deep learning | 0.957 |

## 5. Conclusion

This paper focuses on the dynamic analysis of malicious samples. We proposed a combination classification architecture that combines machine learning and deep learning. A new API sequence process algorithm AMHARA is proposed in this paper as well. It is efficient and easy to implement. We also study the sequence data preprocessing method to remove the redundant data. Experiments show that the integrated classifier has a better performance than the separate machine learning or deep learning. Our method can identify unknown malicious samples well.

## Acknowledgements

This research work was supported by National Natural Science Foundation of China (Grant No. 61472046), Key Lab of Information Network Security of Ministry of Public Security (No.C17607), Seed Funds of Beijing Association for Science and Technology and NSFOCUS Kunpeng Finds.

## References

[1] Xiaozhou Wang, Jiwei Liu, Xueer Chen, "Say No to Overfitting," University of Pittsburgh,2015.
[2] Lipton Z C, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning[J]. arXiv preprint arXiv:1506.00019, 2015.
[3] HUANG Quanwei. Malicious Executables detection based on N-Gram System call Sequences [D]. Harbin Institute of Technology. 2009
[4] LIU YANG. Employing the Algorithms of Random Forest and Neural Networks for the Detection and Analysis of Malicious Code of Android Applications[D]. Beijing Jiaotong University. 2015
[5] YANG Hong-yu, XU jin. Android malware detection based on improved random forest[J]. Journal on Communications, 2017,(04):8-16.
[6] ZHANG Jiawang, Li Yanwei. Malware detection system implementation of Android application based on machine learning[J]. Application Research of Computers,2017,(06):1-6.
[7] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection[J]. Information Sciences, 2013, 231: 64-82.
[8] Ravi C, Manoharan R. Malware detection using windows API sequence and machine learning[J]. International Journal of Computer Applications, 2012, 43(17): 12-16.
[9] LIAO Guohui, LIU Jiayong. A Malicious Code Detection Method Based on Data Mining and Machine Learning[J]. Journal of Information Security Research, 2016,(01):74-79.（in Chinese）
[10] Dahl G E, Stokes J W, Deng L, et al. Large-scale malware classification using random projections and neural networks[C]//Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013: 3422-3426.
[11] Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features[C]//Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on. IEEE, 2015: 11-20.
[12] Kolosnjaji B, Zarras A, Webster G, et al. Deep Learning for Classification of Malware System Call Sequences[C]//Australasian Joint Conference on Artificial Intelligence. Springer International Publishing, 2016: 137-149.
[13] Tobiyama S, Yamaguchi Y, Shimada H, et al. Malware Detection with Deep Neural Network Using Process Behavior[C]//Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual. IEEE, 2016, 2: 577-582.
[14] Pascanu R, Stokes J W, Sanossian H, et al. Malware classification with recurrent networks[C]//Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015: 1916-1920.
[15] Tensorflow. https://www.tensorflow.org
[16] Virusshare https://virusshare.com
[17] VirusTotal. http://www.virustotal.com
[18] Scikit-Learn. http://scikit-learn.org/