API Call Based Ransomware Dynamic Detection Approach Using TextCNN

Bin Qin
Shenzhen University Information Center
Shenzhen, China
qinbin@szu.edu.cn

Yalong Wang*
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
yalongwong@foxmail.com

Changchun Ma
Innovation Insitute
Sangfor Technologies Inc.
Shenzhen, China
machangchun@sangfor.com.cn

***Abstract*—In recent years, the number of ransomware attacks has grown exponentially. Ransomware detection for Windows platforms has become extremely important. In the field of malware detection, the API call is used in many methods, and the API call sequence can be regarded as a sentence in the language. In this paper, the TextCNN model in the Natural Language Processing field is used to detect ransomware and the chunk-based max-pooling is used to improve the pooling layer of the TextCNN model. This paper proposes a Dynamic Ransomware Detector based on the improved TextCNN(DRDT). DRDT is trained with ransomware and benign software's API call sequences. Then API call sequences from unknown programs can be sent to DRDT to determine whether the files are ransomware. The experimental result shows that the detection speed of DRDT is faster than traditional methods, with the accuracy and F1 score of 0.959.**

***Keywords-API; ransomware; TextCNN; natural language processing; deep learning Introduction*;**

I. INTRODUCTION

Ransomware is a kind of malicious software to prevents users from using their computers or data. The user can only recover the locked computer after paid-up a ransom. According to Trend Micro research [1], the new family of ransomware increased by nearly 400% in 2016.

Existing solutions for detecting ransomware rely on static detection technology, but ransomware makes itself increasingly difficult to be detected by using code obfuscation and deformation techniques. The dynamic detection technology can monitor the ransomware in real-time, which overcomes the shortcomings of static detection technology that can not detect ransomware with obfuscation and deformation techniques. Ransomware has clear behavioral characteristics, including file traversal, file hiding, registry auto-start settings, and network interaction with C&C servers where encryption keys are stored, etc. A program running on Windows wants to achieve its function, it needs to call the Windows API(Application Programming Interface). If the API call sequence can be regarded as a sentence in the language, the text classification technology can be used to detect ransomware. In the field of text classification, the TextCNN model has a good effect, and in order to satisfy the work scenario, we use chunk-based max-pooling to improve the pooling layer of TextCNN. By using above methods， a Dynamic Ransomware Detector based on the improved TextCNN(DRDT) can be constructed. In summary, the following innovations was made in this paper:

- By using the API call sequence, this paper proposes a method to construct DRDT, which is simple and faster than other public reported methods.
- To obtain more semantic information features from the API call sequence, we improved the pooling layer of TextCNN by using chunk-based max-pooling.

The structure of this paper is as follows. In Section II, recent literature will be reviewed. In Section III, it will introduce the API call sequence, Natural Language Processing (NLP) and the TextCNN model. In Section IV, our method will be discussed in detail, including the extraction of API sequence, the pooling layer's improvement of TextCNN, and the structure of our TextCNN-based classifier. In Section V, we will evaluate DRDT and compare it with other similar methods. Finally, the conclusion and future work are provided

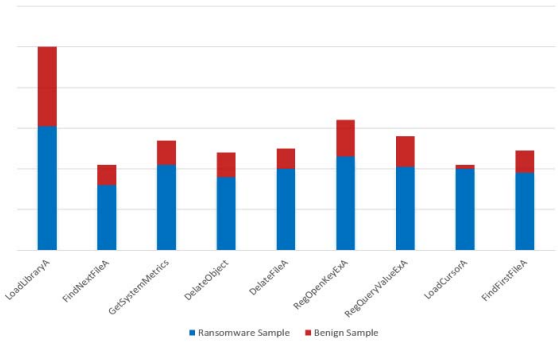

Figure 1. The comparison of some API call frequencies between a ransomware sample and a benign sample in our training data.

978-1-7281-6499-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICBAIE49996.2020.00041

II. Related Works

For dynamic detection of ransomware, scholars have done a lot of research:.

Greg Cusack et al. used Random Forest to detect ransomware by its' abnormal interaction with the C&C server [2]. Their classification model achieves a detection rate of over 0.86, while maintaining a false negative rate under 0.11. Aragorn Tseng et al. constructed a deep neural network and trained the perceptron with critical payloads selected from packets which were extracted from real network traffic, the model can perform about a 93% accuracy rate from their experiment results.

Daniele Sgandurra et al. used Randomized Logistic Regression(RLR) to detect abnormal registry operations and file operations of ransomware with an AUC of 0.995 [3]. Amin Kharaz and Sajjad Arshad utilizes filesystem's activities to classify ransomware from other types of malware with a 96.3% detection rate [4].

Sumith Maniath et al. used the LSTM neural network to detect the ransomware and obtained a testing accuracy of 96.67% [5]. They regarded the API call sequence of ransomware and benign software as text to detect ransomware. Nikolai Hampton summarized the similarity of ransomware's behaviors on the Windows platform and affirmed the practical effect of API call sequence on ransomware detection system [6]. Sajad Homayoun et al. used the LSTM and CNN neural network to identify the ransomware family through API call sequence, and achieved a true positive rate of 0.972 and a false positive rate of 0.027 [7].

Similar to Sumith Maniath et al, this paper proposes a method using the API call sequence to detect ransomware and determine unknown files as ransomware or benign software. A high detection rate and less time consumption are crucial for ransomware detection. However, the traditional detection methods take a long time, this will be solved in our method.

III. Background

A. API Call Sequence

Each specific API call is the exact operation performed by malware or benign software on the running state of the system, such as creating, reading, writing and, deleting files or modifying registry keys. Ransomware and benign software will have their specific API call patterns or unique order of calls. Fig. 1 shows the comparison of some API call frequencies between a ransomware sample and a benign sample in our training data.

B. Natural Language Processing (NLP)

NLP belongs to a subfield of artificial intelligence. It is an automatic method to understand and analyze natural language and obtain information by using the machine learning algorithm, including speech recognition, natural language generation, information extraction, text categorization, etc. The computer program communicates with the operating system through API calls. This communication is similar to how people use language to communicate with others. Each type of program has a specific API call pattern, thus NLP can be applied to the field of malware detection [8].

C. TextCNN

TextCNN was proposed by Yoon Kim in 2014 [9], and has 4 layers: embedding layer, convolution layer, pooling layer, and fully connected layer. It is the application of the convolutional neural network (CNN) in text classification. TextCNN trains on the pre-trained word vectors, and extracts the information in sentences by different convolutional kernels to capture features. TextCNN is simple and fast. It often has a good effect on the text classification field. Fig. 2 shows the network structure of TextCNN.

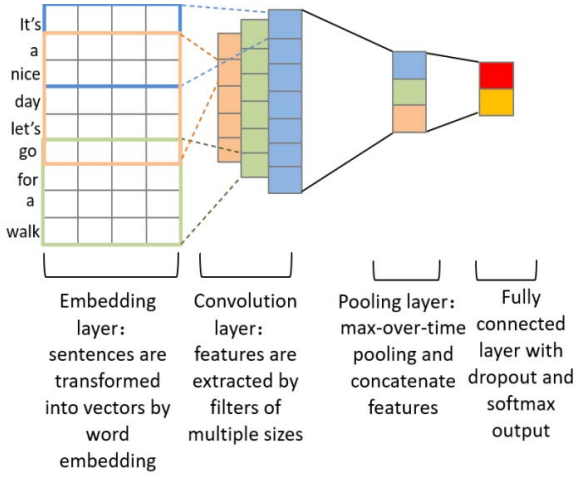

Figure 2. The structure of the TextCNN mode

IV. Proposed Method

The method focuses on using TextCNN to detect ransomware. Fig. 3 shows our system design.

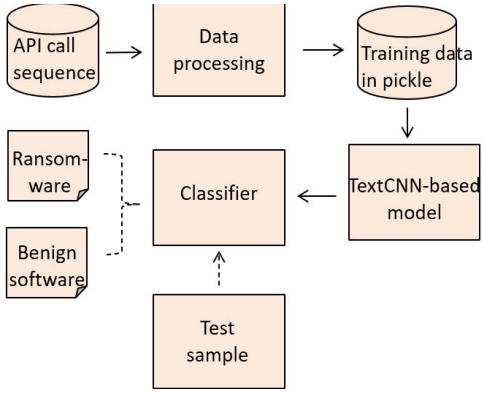

Figure 3. The structure of DRDT system.

A. Extraction of API Call Sequence

Sandbox is a virtual system program that creates an independent operating environment and monitors program behavior. Computer security practitioners often use sandboxes to test high-risk software.

Once the malware is submitted to the sandbox, the sandbox will execute the malware inside the virtual machine, monitor the actions of the process and its child processes, and record the behavior of API calls, registry modifications, file operations, etc. and save in JSON files. The extraction process of the API call sequence is shown in Fig. 4.

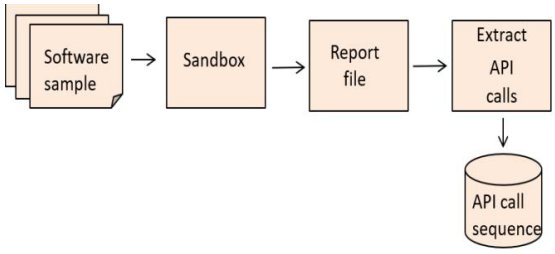

Figure 4. The extraction process of API call sequence.

B. *Structure of Our Improved TextCNN Model*

The Convolution layer of TextCNN extracts features in the text through a one-dimensional convolution. The words in the text are represented by k-dimensional word vectors, then $x_i \in R^k$ represents the i-th word in the text. A sentence of length n (padded where necessary) is expressed as:

$$X_{1:n} = [X_1, X_2, \dots, X_n] \tag{1}$$

The convolution kernel $w \in R^{hk}$ is used to perform a convolution operation to generate a new feature, and h is the size of the convolution kernel window. For example, a feature y_i is generated from a window of words $X_{i:\ i+h-1}$ by:

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

In this formula, $b \in R$ is a partial term, and f is a non-linear function. The convolution kernel w with a window size h performs on the sentence $\{X_{1:\ h}, X_{2:\ h+1}, X_{3:\ h+2}, \dots X_{n-h+1:\ n}\}$ and produces a feature map:

$$y = [y_1, y_2, y_3, \dots y_{n-h+1}] \tag{3}$$

After extracting features through convolution, TextCNN uses a method called max-over-time pooling to obtain the maximum value of y, as shown in Fig. 5a. This method is easy to implement. However, there are some problems with this pooling method used by TextCNN:

- most of the information in the sentence is lost,
- the word order information is missing.

Zhang J et al. proposed a new pooling method chunk-based max-pooling algorithm, to retain more semantic information of the sentence and keep the order information as well[10]:

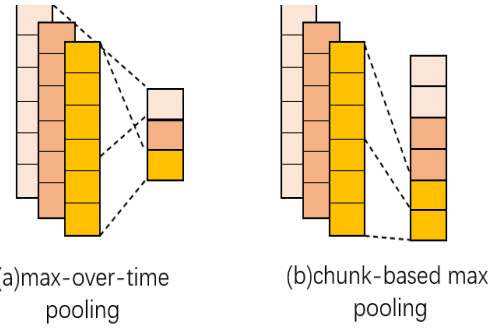

Figure 5. Comparison of max-over-time pooling and chunk-based max pooling.

$$y^C = chunkMax\{[y_1, y_2, y_3, \dots y_{n-h+1}]\}$$

$$= [y_{c1}, y_{c2}, y_{c3}, \dots y_C] \tag{4}$$

In this formula, C is the predefined parameter, representing the number of chunks(e.g. C=2 in Fig. 5b).

The new model uses chunk-based max-pooling to improve the pooling layer. In this way, more text features obtained by convolution will be preserved.

Intaking the advantages of the above methods, we used TextCNN to build the detector and improved the pooling layer using chunk-based max-pooling. More semantic information were extracted by using multiple convolution kernels and multiple sizes of the chunk. Also, we used dropout to prevent overfitting.

The structure of our TextCNN-based classifier is shown in Fig. 6.

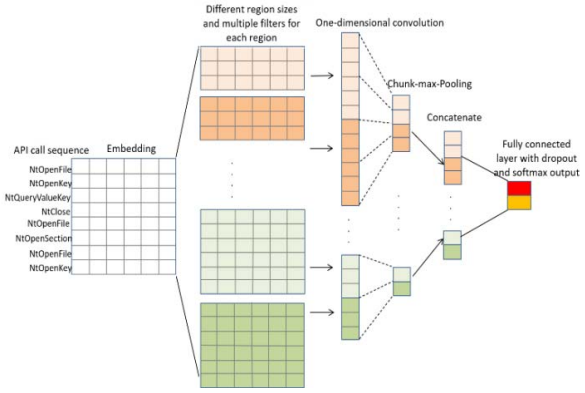

Figure 6. The structure of our TextCNN-based classifier.

V. IMPLEMENTATION

The DRDT was built by Keras, and the experimental GPU used NVidia Quadro GP100 with 16G memory.

A. Data set

The data set used 1000 ransomware samples and 1000 benign software samples provided by Sangfor Technologies Incorporation. We used 80% of the total 2000 samples to train the TextCNN network and the remaining 20% for testing.

B. Experiment

In this experiment, we used pre-trained word vectors for word embedding and multiple convolution kernels of different sizes to extract features. Also, different dropout rates and fully connected layer parameters were tried to find the best parameter setting.

C. Eevaluation

For classification problems, the confusion matrix can comprehensively evaluate the performance of the model, and it can derive many indicators, such as accuracy, precision, recall, and F1 score. The F1 score can comprehensively reflect the detection effect for the ransomware samples, and the accuracy can measure the comprehensive recognition effect of the model on benign software and ransomware samples. Thus we used the F1 score and accuracy to evaluate the model. Similarly, time consumption is one of the most important evaluation criteria for malware detection. Therefore, the indicators derived from the confusion matrix and time consumption were used to evaluate different models. Table I shows the definition of the confusion matrix. The evaluation indicators derived from the confusion matrix are as follows:

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \tag{5}$$

$$Precision = \frac{TP}{(TP+FP)} \tag{6}$$

$$Recall = \frac{TP}{(TP+FN)} \tag{7}$$

$$F1-Score = \frac{2*Precision*Recall}{(Precision+Recall)} \tag{8}$$

Table II shows the performance of different models (using the same experimental platform and data). The experimental result shows that DRDT has better performance and shorter time consumption.

TABLE I. CONFUSION MATRIX

Real situation	Prediction results	
	Positive	***Negative***
Positive	True Positive(TP)	False Positive(FN)
Negative	False Positive(FP)	True Negative(TN)

TABLE II. EXPERIMENTAL RESULTS OF DIFFERENT MODELS

Model	Accuracy	F1-score	Time consumption
LSTM	0.887	0.884	19.52s
CNN-LSTM hybrid	0.900	0.899	13.50s
TextCNN	0.928	0.926	0.80s
DRDT	**0.959**	**0.959**	**1.59s**

VI. CONCLUSION

Considering the semantic information contained in the API call sequence as natural language and the outstanding performance of TextCNN in the field of text classification, we used chunk-based max-pooling to improve the pooling layer of TextCNN and constructed a Dynamic Ransomware Detector based on the improved TextCNN(DRDT).

Experimental result shows that DRDT has a better performance, the accuracy reaches 0.959 and the F1 score reaches 0.959. Also, time consumption is one of the most important evaluate indicators. Although LSTM and other models can capture the context information in API sequence, it can't be used in large-scale sample analysis tasks, because of the longer time computation. The LSTM and CNN-LSTM hybrid ransomware detection model takes more than 13.5s to judge samples that are ransomware or benign software. DRDT has a great advantage over other models in this indicator and compared with the basic TextCNN model, the detection effect is improved. This work will facilitate large-scale automated analysis of ransomware samples to filter out benign executable files.

To make the detector have a higher detection effect, in the future, we will consider using other methods to improve the convolutional layer of TextCNN.

ACKNOWLEDGMENT

Thanks to Sangfor Technologies Incorporation for providing the experimental data, and the joint graduate training base between Shenzhen University and Sangfor Technologies Incorporation for providing the experimental environment.

REFERENCES

[1] "Trend MIcro," 6 Nov 2016. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/research- and-analysis/predictions/2017. [Accessed 2 2 2017].

[2] Cusack G, Michel O, Keller E. Machine learning-based detection of ransomware using sdn[C]//Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization. ACM, 2018: 1-6.

[3] Sgandurra D, Muñoz-González L, Mohsen R, et al. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection[J]. arXiv preprint arXiv:1609.03020, 2016.

[4] Kharaz A, Arshad S, Mulliner C, et al. {UNVEIL}: A Large-Scale, Automated Approach to Detecting Ransomware[C]//25th {USENIX} Security Symposium ({USENIX} Security 16). 2016: 757-772.

[5] Maniath S, Ashok A, Poornachandran P, et al. Deep learning LSTM based ransomware detection[C]//2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE). IEEE, 2017: 442-446.

[6] Hampton N, Baig Z, Zeadally S. Ransomware behavioural analysis on windows platforms[J]. Journal of information security and applications, 2018, 40: 44-51.

[7] Homayoun S, Dehghantanha A, Ahmadzadeh M, et al. DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer[J]. Future Generation Computer Systems, 2019, 90: 94-104.

[8] Tran T K, Sato H. NLP-based approaches for malware classification from API sequences[C]//2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES). IEEE, 2017: 101-105.

[9] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

[10] Zhang J, Zhang D, Hao J. Local translation prediction with global sentence representation[C]//Twenty-Fourth International Joint Conference on Artificial Intelligence. 2015.