Journal of Quantum Computing
DOI:10.32604/jqc.2021.017365
Article

Malware Detection Based on Multidimensional Time Distribution Features

Huizhong Sun[1], Guosheng Xu[1,*], Hewei Yu[2], Minyan Ma[3] , Yanhui Guo[1] and Ruijie Quan[4]

[1]School of Cyberspace Security, Beijing University of Posts and Telecommunication, Beijing, China
[2]Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing, China
[3]Zhejiang Branch of National Computer Network Emergency Response Technical Team/Coordination Center of China, Hangzhou, China
[4]Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia
[*]Corresponding Author: Guosheng Xu. Email: guoshengxu@bupt.edu.cn

Received: 16 March 2021; Accepted: 26 May 2021

Abstract: Language detection models based on system calls suffer from certain false negatives and detection blind spots. Hence, the normal behavior sequences of some malware applications for a short period can become malicious behavior within a certain time window. To detect such behaviors, we extract a multidimensional time distribution feature matrix on the basis of statistical analysis. This matrix mainly includes multidimensional time distribution features, multidimensional word pair correlation features, and multidimensional word frequency distribution features. A multidimensional time distribution model based on neural networks is built to detect the overall abnormal behavior within a given time window. Experimental evaluation is conducted using the ADFA-LD dataset. Accuracy, precision, and recall are used as the measurement indicators of the model. An accuracy rate of 95.26% and a recall rate of 96.11% are achieved.

Keywords: System call; sequence; malware; neural network

1 Introduction

With the advent of the big data era, the types and quantities of malware have exploded. Traditional host-based intrusion detection systems are thus facing unprecedented challenges [1,2]. At the same time, as the convenience and functionality of mobile devices continue to increase, the use of mobile software has also become the norm. Statistics indicate that in the third quarter of 2017, 16 million users were attacked by mobile malware; the number of mobile workers is expected to reach 1.75 billion by 2020 [3]. However, users generally have little understanding of malicious software, and their ability to recognize abnormal software behaviors is limited. Such abnormal software, through various means and methods, seriously threaten the data privacy and property security of users and enterprises [4]. Therefore, a software is urgently needed. An abnormal behavior analysis system can detect the abnormal behavior of software in real time and prevent losses.

Good data features are the prerequisite for the development of intrusion detection systems based on machine learning. System calls are the most widely used features. A large number of intrusion detection systems based on machine learning use system call names to build detection models, and a few studies have built detection models on the basis of system call parameters. Although detection models based on call names or parameters have achieved good results, their detection performance requires improvement. Therefore, the simultaneous use of system call names, parameters, and return values can be considered to inherit their advantages and improve the performance of intrusion detection systems. However, no research has explored the idea of combining applicable system call names, parameters, and return values to improve the performance of systems' detection modules. This deficiency may be attributed to the difficulty of processing a large number of complex system call parameters and return values and to the time overhead involved. Therefore, we construct a modern intrusion detection dataset that contains complete system call information and processes data to complete system call vectorization.

2 Related Work

At present, applying deep learning to the field of intrusion detection is an important topic. Deep learning has achieved good results in the fields of speech recognition, image recognition, and natural language processing. Deep learning can extract high-level abstract features from original features, and it does not require feature selection based on experts' experience. Consider the advantages of deep learning, scholars at home and abroad have attempted to apply deep learning technology to the field of network security. Creech et al. proposed a new host-based abnormal intrusion detection method, which uses a discontinuous system call mode to improve the detection rate while reducing the false alarm rate [5]. Marteau [6] proposed a new sequence similarity measure called "covering similarity," which uses a suffix tree data structure to calculate coverage similarity, evaluate the minimum number of subsequences required to construct a complete coverage of a given sequence, and distinguish between normal sequences and attack sequences. Chan [7] classified system call sequences by using a linear support vector machine optimized by stochastic gradient descent and reported excellent results. Aafer et al. proposed the idea that the bytecode of an Android application contains information that can be used to describe its behavior [8]. Dimjašević et al. proposed a fully automated method for Android malware detection that is based on machine learning system call tracking and the classification of application libraries [9]. Saracino et al. proposed a behavior-based multilevel Android malware detection system called MADAM [10]. Vidal et al. proposed an Android malware detection system [11] that collects core system call sequence data during the startup phase of Android applications.

Although the behavior sequences of some malware in a short period may be normal for deep learning, they can present malicious behavior within a certain time window. To detect such behaviors, we conduct statistical analysis and extract a multidimensional time distribution feature matrix. This matrix mainly includes multidimensional time distribution features, multidimensional word pair correlation features, and multidimensional word frequency distribution features. A multidimensional time distribution model (MTDM) based on neural networks is built to detect overall abnormal behavior within a given time window.

3 Design and Construction of Multidimensional Time Features

This section mainly introduces the construction and extraction of multidimensional behavior features and provides data feature support for the subsequent building and training of neural network models with multidimensional time distribution.

3.1 Multidimensional Feature Design

The multidimensional behavior characteristics proposed in this work mainly refer to multidimensional time distribution characteristics, multidimensional word pair correlation characteristics, and multidimensional word frequency distribution characteristics. We define the total number of word pairs in the sample set as k, the timestamp of any system call word as t_i, the cumulative time interval between a certain system call word and any system call word as WT_i, and the time point value function as f. Then,

$$WT_i = \sum_j^k (t_i - t_j) \tag{1}$$

$$f_m =< \min | \max | mean | median | ... > \tag{2}$$

We define the dictionary size of the system call as n, the dimension of time distribution as m, and the time distribution set as T_m.

$$T_m = \{f_m(WT_i) \,|\, i, j \in n\} \tag{3}$$

We define the threshold set of the time distribution points as P. Then,

$$P = \{f_m(T_m)\} \tag{4}$$

Table 1: Statistical table of the time interval distribution of normal sample

Type	Max	Min	Mean	Median
Max Set	378.293	0	57.606	3.880
Min Set	38.481	0	0.204	0
Mean Set	38.481	0	0.285	0.00125
Median Set	38.481	0	0.210	0.00013

In this work, the time interval set of all system calls of the normal sample is calculated according to the maximum value Max, minimum value Min, average value Mean, and median Median to obtain the maximum value set Max Set, minimum value set Min Set, average value set Mean Set, and Median Set Median Set, respectively.

To analyze the distribution of each set, we calculate the maximum value Max, minimum value Min, average Mean, and median Median of each set (Tab. 1).

The time interval collection of all system calls of the abnormal sample is the same as that in Tab. 1. Statistical analysis is conducted according to the maximum value Max, minimum value Min, average value Mean, and median Median (Tab. 2).

Table 2: Statistical table of time interval distribution of abnormal sample

Type	Max	Min	Mean	Median
Max Set	349.926	0	25.588	0.0715
Min Set	0.0079	0	5.9533	0
Mean Set	5.7166	0	0.0674	0.00025
Median Set	1.071	0	0.0101	0.00012

Through the statistical analysis of Tabs. 1 and 2, we can find that the time interval distributions of the system call of the normal sample and abnormal sample are different. The maximum value of the time interval between the system calls of the normal sample, Max Set, is generally much larger than that of the abnormal sample, Max Set. The minimum value of the time interval between the system calls of the abnormal sample, Min Set, is generally greater than that of the normal sample. The minimum value of the time interval between system calls, Min Set, is relatively small. Similarly, the distributions of the average Value Set Mean Set and the Median Set Median Set of the normal sample and those of the abnormal sample obviously differ. Therefore, the multidimensional time distribution feature used by the MTDM in this work is one of the effective features to distinguish between normal samples and abnormal samples.

The multidimensional word pair association feature refers to the association feature between different system call words. That is, any system call word will be adjacent to multiple system calls and non-adjacent to other system calls. Therefore, the MTDM is used to monitor multidimensional behavior characteristics within a period of time to evaluate abnormal behavior during this period. The full amount of system call data is used, thereby benefiting the extraction of the subsequent multidimensional word frequency distribution features.

The dictionary size of the system call is defined as *n*, and any set of word pairs is WP_k. Then,

$$WP_k = \{(w_i, w_j) \mid i, j \in n\} \tag{5}$$

The multidimensional word frequency distribution feature is the word frequency distribution feature of all system calls. That is, the collection of word frequency features is performed by monitoring the system call within a period of time. The multidimensionality of the multidimensional word frequency distribution feature refers to the multidimensionality of the time interval between different system call words. The multidimensional word frequency distribution feature can effectively capture the quantitative information of different time intervals between different system calls, that is, the quantitative information of the correlation between a certain system call and any other system call words.

This study builds a neural network model with a multidimensional time distribution. The MTDM adopts multidimensional time distribution features, multidimensional word pair correlation characteristics, and multidimensional word frequency distribution characteristics. It can effectively learn the time dimension, word pair correlation dependence, and word frequency distribution. It can also enhance the detection of abnormal behaviors of Android malware.

3.2 Multidimensional Time Eigenvectors

Normal Android software and abnormal Android software invoke system calls at different speeds and frequencies at runtime, and multidimensional features present distribution differences. This section establishes the feature vector by counting the frequency distribution of the time interval between adjacent system call words in the system call sequence in each time range.

We define a sequence as *S* and the dictionary defining system calls as *D*. Then,

$$S = \{w_1, w_2, w_3 ... w_n \mid w_i \in D\} \tag{6}$$

The time series of system calls corresponding to the definition sequence *S* is T. Then,

$$T = \{t_1, t_2, t_3 ... t_n\} \tag{7}$$

The time interval sequence between system call words corresponding to the definition sequence *S* is *G*. Then,

$$G = \{t_1 - t_0, t_2 - t_1, t_3 - t_2 \ldots t_n - t_{n-1}\} \tag{8}$$

The calculation of the time gap between different system call word pairs is shown in Tab. 3.

Table 3: Time gap calculation matrix between different system call word pairs

Word	w_1	w_2	w_3	…	w_n
Time	t_1	t_2	t_3	…	t_n
Gap	t_1-t_0	t_2-t_1	t_3-t_2	…	t_n-t_{n-1}

Assume that the dimension of the time gap frequency distribution is M. Then, the value of dimension M will also be different in different data scenarios. The appropriate M can be set by calculating the distribution of the time gap between words. The index number of a system call is Idx, and the time interval distribution between word pairs (w_i,w_j) is *TGD*(w_i,w_j), as shown in Tab. 4.

Table 4: Time interval distribution of word pairs

Idx	1	2	3	…	n
Time range	<1 ms	1~2 ms	2~3 ms	…	>n
TGD (w_i,w_j)	5	2	1	…	9

The value of each *TGD* (w_i,w_j) represents the frequency of occurrence of the system call word pair (w_i,w_j) in this time interval distribution. On the basis of the statistics and analysis of the time interval distribution of all system calls in the normal and abnormal samples, we set $M = 4$ of the MTDM.

In a real Android running environment, the order of system call words is irregular. For example, the word w_i may appear before or after the word w_j, and the words w_i and w_j may be the same system call word. To effectively describe the correlation dependence of this type of word pair (w_i,w_j) and record the distribution characteristics of multidimensional behavior features, we use the adjacency matrix in constructing the feature vectors in this work.

Assume that the dictionary size of the entire dataset is k. Then, the eigenvectors of each sample can be represented by a three-dimensional adjacency matrix, as shown in Tab. 5.

Table 5: Adjacency matrix of multidimensional time distribution features

w_n	$TGD(w_1,w_n)$	$TGD(w_1,w_n)$	…	$TGD(w_n,w_n)$
…	…	…	…	…
w_2	$TGD(w_1,w_2)$	$TGD(w_2,w_2)$	…	$TGD(w_n,w_2)$
w_1	$TGD(w_1,w_1)$	$TGD(w_2,w_1)$	…	$TGD(w_n,w_1)$
	w_1	w_2	…	w_n

If no adjacent relationship exists between any pair of words (w_i,w_j), then the default value of $TGD(w_i,w_j)$ can be set to [0,0,0…0].

The feature vector designed in this work not only retains the frequency information of system call words but also stores the information of the correlation between system call words and the time interval distribution between adjacent system call words. Information about system call behavior differences in normal and abnormal samples can be stored to improve the accuracy of subsequent model training.

3.3 Time Distribution Model

The multidimensional behavior characteristic constructed in this work is a three-dimensional adjacency matrix. This adjacency matrix is similar to the three channels of the picture in the data dimension, and the time interval distribution of each system call corresponds to a channel number. In view of the great success of CNNs in image recognition [12], we adopt a CNN to perform multidimensional feature learning.

CNNs are calculated by multilayer convolutions and use nonlinear activation functions, such as ReLU and tanh. The convolution of each layer is shared by the parameters of the convolution kernel filters, and local iterative learning is performed according to the set number of sliding steps. During the training phase, the CNN automatically learns the parameter values of the filters on the basis of the provided training set.

In image classification, the CNN in the first layer of the edge can be detected from the original pixels [13], and that in the second layer can find a number of simple shapes. These shapes are used on a higher layer to find similar high-level characteristics, such as hull, that is, “from pixel to edge, from edge to shape, and from out of shape into a more complex object”. For the multidimensional behavior characteristics based on system calls, the CNN is used to learn local behavior characteristics first and is then applied to a higher level of feature learning. Multiple convolutional layers can be used to obtain a deep feature map. The last layer involves the use of these high-level features for classification.

The specific architecture of the designed MTDM based on neural networks is shown in Fig. 1.

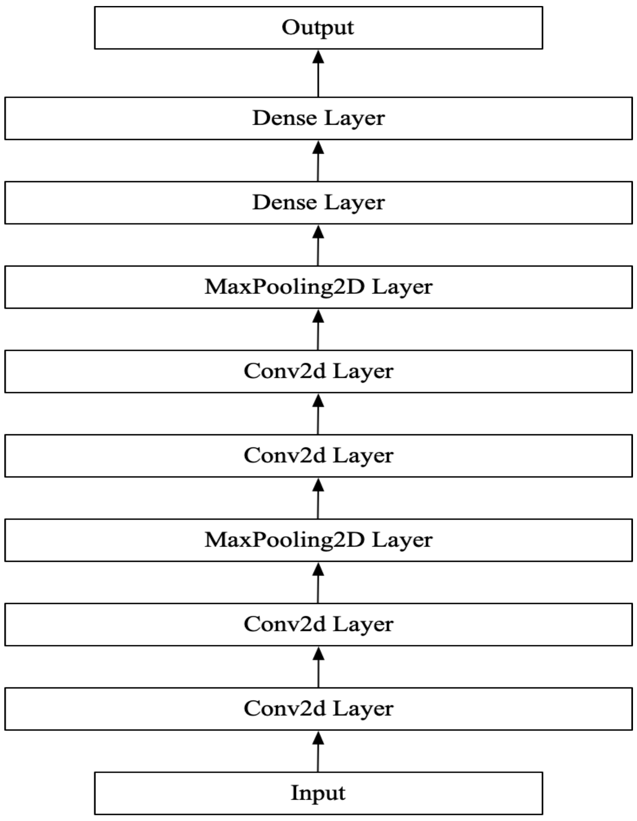

Figure 1: Architectural diagram of MTDM

The MTDM designed in this work includes the input, conv2d layer * 2 of convolutional layer, MaxPooling2D layer of maximum pooling layer, conv2d layer * 2 of convolutional layer, MaxPooling2D layer of maximum pooling layer, MaxPooling2D layer of maximum pooling layer, dense layer * 2 of full connection layer, and output.

4 MTDM Performance Test and Evaluation

4.1 Evaluation Metrics

To evaluate the quality of the proposed model, we use TP, FN, FP, and TN. TP means that the true label of the sample is positive and that the number of samples predicted by the model is positive. FN means that the true label of the sample is positive and that the number of samples predicted by the model is negative. FP means that the true label of the sample is negative and that the number of samples predicted by the model is positive. TN means that the true label of the sample is negative and that the number of samples predicted by the model is negative.

Accuracy denotes the proportion of correct results that is judged by the classification model in the total number of results.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{FN} + \text{TN})} \tag{9}$$

Precision is the proportion of TP results judged by the classification model in the proportion of positive results.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{10}$$

Recall rate is the proportion of the results with TP values judged by the classification model as correct.

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{11}$$

Specificity is the proportion of the results with TN values judged by the classification model as correct.

$$\text{Specificity} = \frac{\text{TN}}{(\text{FP} + \text{TN})} \tag{12}$$

The F1-score is an index used in statistics to measure the accuracy of binary classification models. It considers the accuracy and recall of classification models. The F1-score can be regarded as a harmonic average of model accuracy and recall. Its maximum value is 1, and its minimum value is 0.

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{13}$$

4.2 Dataset

To verify the model's malware detection based on multidimensional time distribution characteristics, we use the ADFA-LD dataset [5,14,15] of intrusion detection released by the Australian National Defence Academy for testing and comparative evaluation.

The dataset is divided into three subsets: training set, attack set, and verification set. The training set and verification set are system calls generated by the operation of benign samples, and the attack set collects the data of eight malicious samples. Each data file records the system call sequence within a period of time, and each system call is represented by a number. The basic composition of the specific ADFA-LD dataset is shown in Tab. 6.

Table 6: Basic composition of ADFA-LD dataset

Type	Nums	Label	Use type
Training	833	Normal	Normal
Validation	4373	Normal	Normal
Hydra-FTP	162	Abnormal	Abnormal
Adduser	91	Abnormal	Abnormal
Java-Meterpreter	125	Abnormal	Abnormal
Meterpreter	75	Abnormal	Abnormal
Hydra-SSH	148	Abnormal	Unknown-abnormal
WebShell	118	Abnormal	Unknown-abnormal

To verify the learning ability of the model against unknown attacks, we use only four attack types, namely, Hydra-FTP, Adduser, Java-Meterpreter, and Meterpreter, as the negative samples of the training set in the dataset processing stage. This type of attack is regarded as an unknown attack type, and the unknown attack test set is obtained after cross deduplication, that is, a completely unknown malicious behavior sequence. Part of the data is extracted from the validation dataset and then separated. After cross deduplication, an unknown positive sample test set is obtained.

4.3 Results and Analysis

The MTDM designed in this work uses the Android dataset for training and verification. The LOSS and ACC changes of the MTDM during the training process are shown in Fig. 2.

Fig. 1 indicates that the LOSS value of the MTDM training set converges as the number of training iterations in one epoch continues to increase and then gradually becomes stable. The LOSS of the validation set is also near the LOSS of the training set as the number of training iterations in one epoch increases. The LOSS value fluctuates up and down and gradually converges. The accuracy of the MTDM also rises and eventually becomes stable.

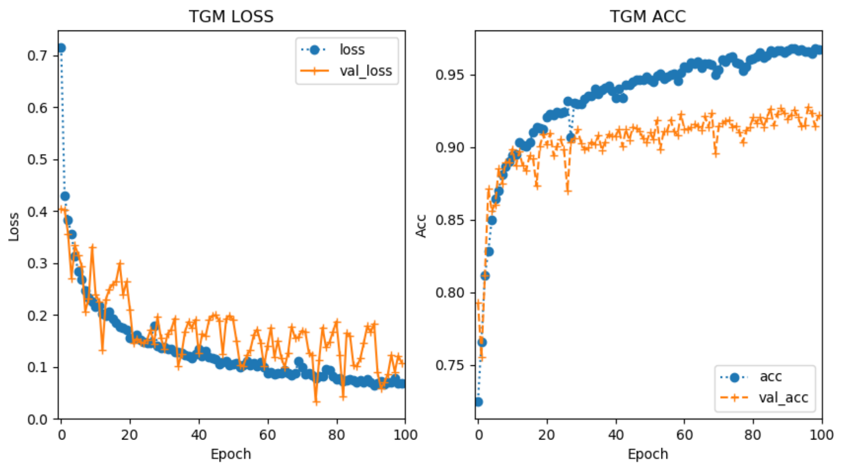

Figure 2: MTDM's LOSS and ACC trends

The specific index results of the MTDM obtained through test evaluation after the completion of MTDM training are shown in Tab. 7.

Table 7: MTDM index evaluation

Model	Precision	Recall	Specificity	F1-score
MTDM	0.95263	0.94988	0.86766	0.96117

5 Conclusion

This work highlights the differences in the time distributions between positive and negative system call samples through statistical analysis. Statistical analysis is used to propose the differences in the multidimensional characteristics of positive and negative samples. These differences are mainly manifested as multidimensional time distribution features, multidimensional word pair association features, and multidimensional word frequency features. On the basis of these differences, the design, construction, and extraction of multidimensional behavior features are proposed. The design ideas, methods, and extraction algorithms for constructing feature matrices are also introduced in detail. On the basis of the aforementioned typical features, a feature matrix is constructed, and the MTDM based on neural networks is built by setting a specific time window to detect abnormal behaviors.

Acknowledgement: We are grateful to the peoples for the support and encouragement.

Funding Statement: This research was supported by the National Key Research and Development Program of China (No. 2017YFB0801900).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] U. Tariq, "Intrusion detection and anticipation system (IDAS) for IEEE 802.15.4 devices," *Intelligent Automation & Soft Computing*, vol. 25, no. 2, pp. 231–242, 2019.

[2] W. Han, Z. Tian, Z. Huang, L. Zhong and Y. Jia, "System architecture and key technologies of network

security situation awareness system YHSAS," *Computers, Materials & Continua*, vol. 59, no. 1, pp. 167–180, 2019.

[3] V., Bourne, *iPass-Mobile-Security-Report.* 2018.

[4] S. Zou, H. Sun, G. Xu and R. Quan, "Ensemble strategy for insider threat detection from user activity logs," *Computers, Materials & Continua*, vol. 65, no. 2, pp. 1321–1334, 2020.

[5] C. Gideon and J. Hu. "A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 807–819, 2013.

[6] P. F. Marteau, "Sequence covering for efficient host-based intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 994–1006, 2018.

[7] K. W. Chan, "NtMalDetect: A machine learning approach to malware detection using native API system calls," *arXiv preprint arXiv:1802.05412*, 2018.

[8] Y. Aafer, W. Du and H. Yin, "DroidAPIMiner: Mining API-Level features for robust malware detection in Android," in *Int. Conf. on Security & Privacy in Communication Systems*, vol. 127, no. 1, pp. 86–103, 2013.

[9] M. Dimjašević and S. Atzeni, "Evaluation of Android malware detection based on system calls," in *Proc. of ACM on Int. Workshop on Security and Privacy Analytics (IWSPA)*, New Orlean, pp. 1–8, 2016.

[10] A. Saracino, D. Sgandurra and G. Dini, "Madam: Effective and efficient behavior-based android malware detection and prevention," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 83–97, 2016.

[11] J. M. Vidal, M. A. Monge and L. J. Villalba, "A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences," *Knowledge-Based Systems*, vol. 150, no. 1, pp. 198–217, 2018.

[12] Z. G. Qu, S. Y. Wu, W. J. Liu and X. J. Wang, "Analysis and improvement of steganography protocol based on bell states in noise environment," *Computers, Materials & Continua*, vol. 59, no. 2, pp. 607–624, 2019.

[13] Z. G. Qu, S. Y. Chen and X. J. Wang, "A secure controlled quantum image steganography algorithm," *Quantum Information Processing*, vol. 19, no. 380, pp. 1–25, 2020.

[14] G. Creech and J. K. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *IEEE Wireless Communications and Networking Conf.*, pp. 4487–4492, IEEE, 2013.

[15] G. Creech, "Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks," University of New South Wales, Canberra, Australia, pp. 6–12, 2014.