VMD: A Visualizable Malware Detection Scheme Based on Multi-dimensional Dynamic Behavior Information

Yulong Cui[1], Zhiwei Zhang[1,2], Zhidong Ma[1], Zehan Chen[1,2], Yuzi Wang[1], and Yulong Shen[1]
[1]School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, 710071, China
[2]Institute of Network Information, Academy of Systems Engineering,
Academy of Military Sciences, Beijing, 100141, China

Enormous traditional malware detection methods have been proposed and they are efficient in detecting known malware, however, these methods usually either are ineffective or have high false positive rate in detecting unknown malware. To improve the performance in unknown malware detection, many novel methods are presented by introducing and employing software behavior visualization, Artificial Intelligence, and other popular technologies. Unfortunately, they are still facing the inadequate data utilization and benchmark dataset lack problems. Therefore, in this paper, we propose an effective malware detection method by utilizing multi-dimensional software dynamic behavior information. To construct the concrete scheme, we combine a mapping mechanism from key parameters of software operation to grayscale images with a malware prediction model. In addition, to our knowledge, we constructed the first publicly accessible software dynamic behavior dataset containing over 800 malicious and 600 non-malicious software behavior images. Finally, the experiment results show that our method can effectively detect unknow malware and outperforms the existing comparative baseline schemes in terms of accuracy, precision, and recall.

Index Terms—**Unknown Malware Detection, Dynamic Behavioral Information, Software Behavior Image Dataset.**

I. Introduction

RECENTLY, a large number of attackers use unknown malware to carry out new cyber attacks, causing huge economic losses to individuals, enterprises and society. In these information security incidents, unknown malware plays an important role and has become a key factor threatening information security. When new cyber attacks are unavoidable, efficient detection of unknown malware has become a key to cyber security.

Traditional malware detection method includes feature and rule-based methods and behavior-based methods [1]-[8]. These detection methods can detect known malware quickly and efficiently. In unknown malware detection, these methods are either ineffective or have a high false positive rate which hinders analysis.

Besides traditional methods, there are various new methods emerging, such as heuristic-based detection methods [9]-[10], deep learning-based detection methods [11]-[17], and visualization-based detection methods [18]-[26], these malware detection methods provide new ideas for unknown malware detection by using deep learning, knowledge reasoning and other AI technologies. Most of these methods can effectively detect unknown malware, but lack a benchmark dataset. Existing public dataset present the following challenges: Data dimension is single. Visualization datasets often contain only static features which is not conducive to unknown malware detection. Some datasets have legal restrictions for being shared.

To solve the above problems, we propose a malware detection method based on multi-dimensional dynamic behavior information. We build a deep learning model to learn dynamic features , and use a trained model to detect unknown malware. In addition, we construct a dynamic software behavior image dataset. It extracts multi-dimensional behavior features from software dynamic reports. We combine dynamic features with grayscale image conversion technology to form a visualization dataset.

Our main contributions in this work are as follows:

1) We propose a malware detection method that combines multi-dimensional dynamic feature extraction with grayscale image processing technology, use the CNN algorithm to detect software behavior, aiming to address the inefficient detection of unknown malware.

2) We construct a multi-dimensional dynamic software behavior image dataset to solve the problem of lack of benchmark datasets. Our dataset contains 848 malicious and 619 non-malicious software images converted from dynamic reports.

3) We conduct experiments to demonstrate the effectiveness of our scheme. Compared with other traditional solutions, our proposed method improves the detection accuracy by up to 15.73%.

The rest of this paper is organized as follows. We review the related work on malware detection methods in Section II and give a rough overview to our system modules in Section III. Further, we build the system model according to the dynamic behavior information preprocess module, behavior information visualization module, and detection module order in Section IV. Then, in Section V, we introduce the dynamic software behavior image dataset and discuss the comparison experimental results in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

In this section, we conduct a detailed investigation and classification of existing malware detection schemes and datasets. Then, we present their details and analyze their shortcomings.

Manuscript received December 17, 2023; revised February 23, 2024.
Corresponding author: Zhiwei Zhang (email: zwzhang@xidian.edu.cn).

A. Malware Detection Methods

In this part, we first analyze feature and rule-based detection methods and behavior-based methods. In addition to the two traditional malware detection methods, we choose some detection methods based on AI technology, such as heuristic-based malware detection methods, deep learning-based malware detection methods, and visualization-based malware detection methods.

1) Feature and Rule-based Malware Detection

Traditional malware detection methods are mainly divided into two categories: feature and rule-based malware detection methods and behavior-based malware detection methods. The feature and rule-based malware detection method can comprehensively analyze and detect malware without executing the software through preset rule conditions. The most representative one is the signature-based malware detection method. Signature-based malware detection methods are widely used in commercial anti-virus. Ma et al. [1] proposed a new malware detection framework based on detection, genetic algorithm (GA) and signature generator. The framework builds the control flow graph of the application and builds an integration model. Zolkipli et al. [2] proposed a bioinformatics technique to generate accurate vulnerability-based features for polymorphic worms. This mode produces more accurate signatures than other vulnerability-based signature generation modes. Borojerdi et al. [3] proposed a detection system based on sequence clustering and alignment. This method mainly targets polymorphic malware and automatically generates signatures based on malicious behavior. Zheng et al. [4] proposed DroidAnalytics, a system based on Android malware analysis. DroidAnalytics can generate signatures for applications and correlate malware with various malware in the database. Signature-based malware detection methods perform fast and efficient in detecting known malware, but are less effective in detecting unknown malware.

2) Behavior-based Malware Detection

Behavior-based malware detection methods monitor the execution process of malware, analyze dynamic behavior, and detect malware based on behavioral characteristics. The method uses monitoring tools to observe the behavior of sample program. Wagener et al. [5] proposed an automated technology for extracting malware behavior from system calls, using alignment technology to identify similarities. Fukushima et al. [6] proposed a method that can simultaneously detect unknown malware and encrypted malware on Windows operating systems. Chandramohan et al. [7] modeled bounded feature space behavior, which limits the number of features for detecting malware. This method helps reduce detection overhead. Das et al. [8] proposed a hardware enhancement architecture using processors and field programmable gate arrays (FPGAs). The architecture implements a frequency concentration model to detect malware. Behavior-based malware detection methods perform well against unknown malware. However, this type of method faces the problem of a high rate of false positives.

3) Heuristic-based Malware Detection

Heuristic-based malware detection methods have been widely used. Ye et al. [9] proposed a correlation classification post-processing technology for malware detection. This technique reduces the number of generated rules by using rule pruning. Bazrafshan et al. [10] proposed a method that can detect new malware. This method generates signature-like labels for suspicious programs. Heuristic detection can be used to detect known malware, but is less effective at detecting unknown malware.

4) Deep Learning-based Malware Detection

The idea of deep learning provides a new way of thinking for unknown malware detection. Dahl et al. [11] proposed the use of random projection and neural networks for large-scale malware classification. Simulation experiment results show that this method performs better in detection effect. Yuan et al. [12] proposed Droid-Sec, which detects malware based on deep learning. This method uses a combination of static and dynamic analysis in the pretraining stage and backpropagation stage. Kumar et al. [13] proposed a new multi-task deep learning architecture for malware classification and detection. Saxe et al. [14] proposed a deep neural network malware detection based on two-dimensional binary program features. Through complementary feature extraction, deep neural network and score calibration, the accuracy of malware detection was improved. Saracino et al. [15] proposed MADAM, a multi-level malware detection system based on Android devices. MADAM simultaneously analyzes and correlates the characteristics of malware at the four levels of kernel, application, user and software package to detect and block malicious behavior. Huang et al. [16] adopted a malware detection method based on Deep Belief Network to better characterize Android malicious applications and improve the effectiveness of detection. Bengio et al. [17] proposed a method for Android malware detection based on machine learning algorithms. This method solves the problem of excessive parameter space in malware detection by using deep structure learning algorithms. Deep learning has improved the efficiency of malware detection effectively, but its application in this area is not mature, and more efficient methods of integration are still being explored.

5) Visualization-based Malware Detection

Convolutional Neural Networks (CNNs) have become a pivotal technology in the realm of computer vision, showcasing remarkable capabilities across various visual tasks. LeCun et al. [18] played a crucial role in pioneering CNN development, introducing the groundbreaking LeNet-5 architecture in 1998. This seminal work laid the foundation for subsequent advancements in image recognition and classification. The introduction of AlexNet by Krizhevsky et al. [19] in 2012 significantly propelled the field forward, achieving unprecedented success in the ImageNet Large Scale Visual Recognition Challenge. Its deep architecture, comprising multiple convolutional layers, underscored the effectiveness of CNNs in handling complex visual data. Following these milestones, more intricate and deeper CNN architectures have been proposed. The VGGNet, presented by Simonyan et al. [20] in 2014, introduced a uniform architecture with small convolutional filters, emphasizing the importance of depth in CNNs. Addressing the challenge of training extremely deep networks, He et al. [21] introduced residual networks (ResNets) in 2015. ResNets employ residual

connections to facilitate information flow through the network, enabling the successful training of models with hundreds of layers. Furthermore, CNNs have demonstrated excellence in object detection tasks. The Region-based CNN (R-CNN) family, introduced by Girshick et al. [22], marked a paradigm shift by amalgamating region proposal networks with CNNs, resulting in enhanced object localization and detection accuracy. In this paper, the CNN algorithm is employed for the binary classification of sample dynamic behavior. Binary classification implies that the malignancy of a sample is determined solely by its dynamic behavior.

Now Visualization-based malware detection method is becoming more popular among all malware detection methods due to its ease of use and infrastructure for synthetic images. This type of method uses image conversion technology to convert malicious binaries or reports into specific images for classification and detection. Nataraj et al. [23] proposed a classification method using standard grayscale image features. Motivated by the observation that for many malware families, the images belonging to the same family appear very similar in layout and texture, this method converts malware binary files into grayscale images and construct malimg dataset. Naeem et al. [24] proposed a model for characterizing malware variants. This model converts malware binary into grayscale images for detection, which achieves high efficiency and accuracy. Su et al. [25] proposed a lightweight new method for detecting DDoS malware in the Internet of Things environment. This method extracts malware image features and uses convolutional neural networks to classify. Many methods have calculated the similarity between different representations of the malware images to perform the classification, as can see in Han et al. [26], where the authors converted executable files into gray-scale images and introduced a similarity technique based on entropy graphs. Experimental results showed that our scheme achieves a 97.9% similarity rate. The disadvantage of existing visualization-based malware detection methods is that the generated image information is relatively simple, makes it difficult to comprehensively analyze the characteristics of the software under test.

B. Public Software Behavior Image Dataset

There is no benchmark dataset for existing malware detection works. Some existing datasets are frequently used in related work. For example, malimg [23], ember [27], malevis [28] and so on. Malimg dataset was constructed by the University of California Vision Research Laboratory. This dataset contains 9339 samples from 25 malware families, which are obtained through network and Windows operating system malware mixing experiments. This dataset does not contain legal code. Ember dataset was built by Hyrum S. et al [27]. It includes features extracted from 1.1M binaries and represents the first large public dataset for machine learning malware detection. Malevis dataset is a corpus containing 26 categories of byte images, involving a total of 14,226 images. However, these datasets are mainly converted from static features of software samples, making it suitable for classifying known malware and difficult to analyze the dynamic behavior of software in the system.

III. VMD: Visualizable Malware Detection Model

The system model introduces the system structure composed of three parts: dynamic behavior information preprocessing module, behavior information visualization module and detection module. The overall system model is shown in Fig.1.

In the system model proposed in this paper, the dynamic behavior information preprocessing module is responsible for collecting reports generated by running samples in the sandbox, formatting the collected reports into standard reports, and inputting the standard reports into the behavior information visualization module. The behavioral information visualization module is responsible for converting the collected reports into specific grayscale images and submitting the grayscale images to the detection module. The detection module is responsible for training detection model and using the pre-trained CNN classifier to detect collected grayscale images. The execution steps of the system model are as follows:

In the first step, put the self-constructed sample library into the pre-configured sandbox for analysis, and put the report generated after analysis into the dynamic behavior information preprocessing module for report formatting. The APIs called by the malware during the execution of the sandbox come from Windows function libraries. During system execution, how to detect abnormal API calls in system is the key to malware detection. Based on this observation, the dynamic behavior information preprocessing module formats input reports from the behavior sample library into standard reports through parameter screening, parameter simplification and deduplication. The formatted standard report mainly includes important API parameter information in the original report, which is used to construct behavioral dataset and perform visual analysis.

In the following step, the standard reports in the behavioral dataset are input into the behavioral information visualization module for visualization operations. Each standard report is processed into a two-dimensional matrix through a text matrixing operation. By converting each row of a standard report into a binary string, all binary strings in each report are combined into a two-dimensional matrix. Based on the intention of visual analysis of data, the behavioral information visualization module needs to convert the generated two-dimensional matrix into a grayscale image, and each element in the two-dimensional matrix corresponds to a pixel in the image, realizing the transformation from standard reports to grayscale images.

Finally, the detection module needs to collect the grayscale images generated in the previous step, and uses the predefined CNN model to train and test. In the convolutional layer, extracting features of each grayscale image, then after being processed of the pooling layer and the connection layer, achieving the goal of training grayscale images. Use the trained CNN model to detect unknown classified samples on the test set to verify the effectiveness of this method in detecting unknown malware.

After using the dynamic behavior information preprocessing module, behavior information visualization module and detection module processing dynamic behavior reports, experiment

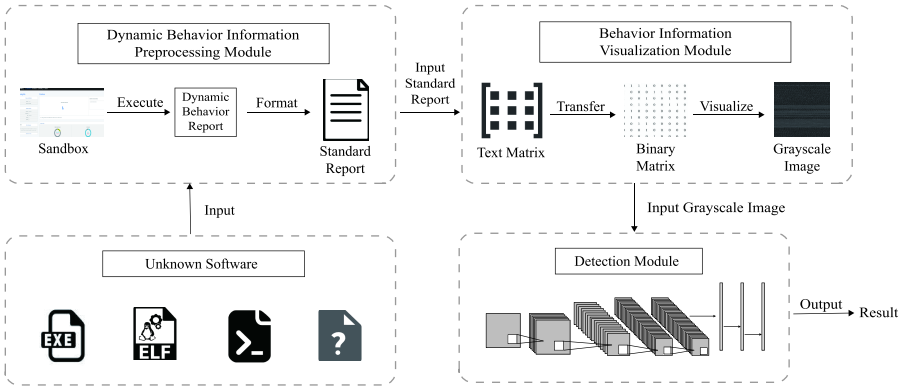

Fig. 1: Malware Detection Model Overview

results verify that the method proposed in this paper can detect malware more accurately.

IV. VMD SCHEME DESIGN

In this section, we first construct a dynamic behavior preprocess module to extract features from dynamic reports. Secondly, in behavior information module, we transfer dynamic features into grayscale images. Finally, we train a CNN model which classify malware and goodware. The detialed process is as follow.

A. Dynamic Behavior Information Preprocess Module

Report Standardization: Original dynamic behavior report is JSON structure. This paper extract important information from original report and convert to prescribed report for later use. This experiment converts dynamic behavior report to a intermediate report containing all the behaviors of the malware sample run in Cuckoo Sandbox. In intermediate report, each line represents an API call, and can be divided into three parts. The first part represents the API type and corresponds to the category field in the original report. The second part of the dynamic behavior report represents the API name. The rest includes relevant parameters during the call, such as path parameters, etc.

```
# pid 3060 tid 2980
registry NtOpenKey"HKEY LOCAL MACHINE"
file GetFileAttributesW 4294967295
"C:\\Windows\\SYSTEM32\\MSCOREE.DLL.local"
"C:\\Windows\\System32\\MSCOREE.DLL.local"
synchronisation NtCreateMutant 0 ""
synchronisation GetSystemTimeAsFileTime
registry NtOpenKey "HKEY_LOCAL_MACHINE"
registry NtOpenKeyEx "HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows
NT\\CurrentVersion\\Diagnostics" 0
system LdrLoadDll "ADVAPI32" 0 "ADVAPI32.dll" 0
system LdrGetProcedureAddress 0 "ADVAPI32" "RegOpenKeyExW"
```

Fig. 2: Textual report

As shown in Fig.2, the first line of the document indicates process identification number (PID) and thread identification number (TID). In the operating system, PID is used as a unique identifier for a process, and TID is used to identify individual threads in the process. The second line of the document contains API information. It includes three parts: API type, where the first API type is ”registry”; API name, in this case ”NtOpenKey”; several important parameters related to the API, for example, in the second line of Fig.2, the root key is designated ”HKEY_LOCAL_MACHINE” and contains system-related information about the local computer, including hardware and operating system details, security data, and the computer’s software settings.

In our experiments, model needs to be trained on standard reports. After extensive experiments, we found that the scheme performs better without considering redundant parameter information. In the formatted report method of this paper, it is necessary to delete the extra parameters except the API type and API name before formatting reports. After deleting the extra parameters, we deduplicate processed reports. We treat adjacent repeated API calls as one record because adjacent duplicate API calls in the report are actually Cuckoo Sandbox calling some duplicate APIs in different guests. A large amount of researches show that these repeated API calls don’t improve the efficiency of detecting unknown malware, but consume a lot of computer resources. After above steps, the original report transformed into standard report, as shown in Fig.3.

B. Behavior Information Visualization Module

Text Matrix: By design, each dynamic behavior is represented as a separate line in a standard report indicating the API type and API name associated with it. Treating each line in the standard report as a single unit, all dynamic behavior logs are traversed to compile a comprehensive lexicon of all observed dynamic behaviors. Establishing a mapping relationship between dynamic behaviors and tag IDs, assigning a unique numerical identifier to each word in the vocabulary. This mapping helps associate specific tags with corresponding dynamic behaviors. In standard reports, in addition to the behaviors identified in dynamic reports, there may be other behavior that is difficult to be characterized. For this type of behavior, a category labeled ”Unknown” has been added to the word. This classification helps capture unknown behaviors and effectively classify them for later analysis. Fig.4 shows an example showing the vocabulary generated from two standard reports.

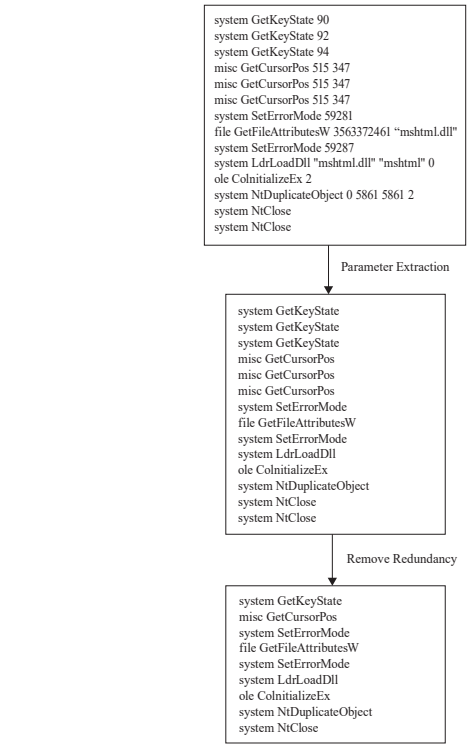

Fig. 3: Report Formatting Process

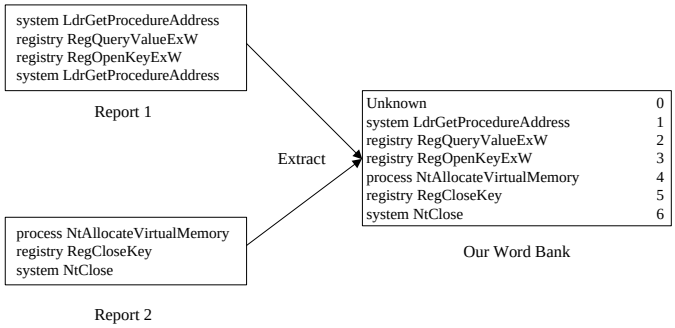

Fig. 4: Word Bank Example

This step converts the entire report into a two-dimensional matrix by converting each row into a one-dimensional matrix. In order to realize the matrix representation of text, each word in the vocabulary is represented as a vector of length 300, and the vector length can be adjusted. Word vectors are randomly initialized and updated continuously during training. For each sample, the lexicon converts the dynamic behavior into the corresponding id sequence, uses a vector to represent each id sequence in the lexicon, and converts the sample into a two-dimensional matrix through this method. The process is shown in Fig.5. It is important to specify a maximum word size during conversion to ensure that all matrices are of consistent size. In case of insufficient sample length, the end of the matrix is padded with zeros. For samples that are too long to handle, this paper uses a direct truncation method to process them.

Behavior Visualization: The visualization process involves converting the dynamic behavior signature file into a grayscale image, where different areas in the grayscale image correspond to specific parts of the dynamic behavior signature file structure. This visualization technique aims to capture similarities in the dynamic behavior exhibited by malware. By representing dynamic behavioral feature files as images, the malware classification and detection problem is transformed into an image classification problem. This paper uses an improved B2M algorithm to convert dynamic behavior feature files into grayscale images [29]. The algorithm follows a specific sequence of steps, as illustrated below:

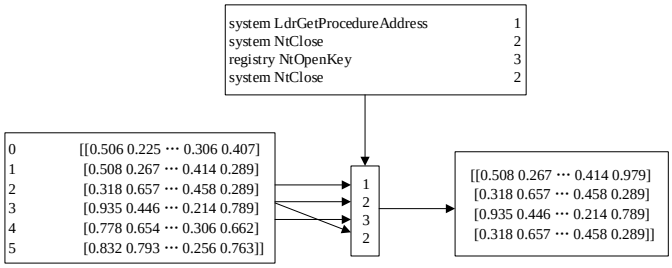

Fig. 5: Transform Example

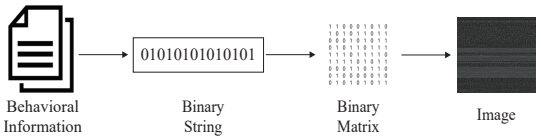

Fig. 6: Behavior Information Visualization

Read the dynamic behavior feature file in binary mode, interpreting each byte as an 8-bit unit. Ensure that each unit's value is within the valid pixel range of 0 to 255, corresponding to the grayscale spectrum where 0 represents black and 255 represents white. Treat each unit as a pixel value for a grayscale image. Process the file size to create a square image with equal width and height, forming a two-dimensional array of pixel values.

The resulting image is a square grayscale representation with pixel intensities ranging from 0 to 255. Fig.7 illustrates an example of the generated image.

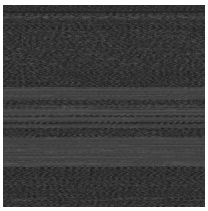

Fig. 7: Image Example

Given the various sizes of the generated dynamic behavior feature files, it is imperative to perform feature extraction and other necessary tasks. Standardization requires resizing all images to ensure uniformity. Only after completing the normalization operation and ensuring identical image sizes, does feature extraction become meaningful as input for the network. This paper employs the bilinear interpolation algorithm for consistent adjustment of image sizes.

C. Detection Module

Numerous studies have leveraged machine learning algorithms to discern the dynamic behaviors feature of samples, showcasing the superior efficiency of such algorithms over manual analyses. Konrad Rieck et al. [30] proposed an incremental approach for behavior-based analysis. This method utilizes the relative order of APIs in the sample as eigenvectors and employs an approximation for clustering and classification algorithms to detect malware samples. Radu et al. [31] applied the random forest algorithm for detecting malicious dynamic behavior, extracting 68-dimensional feature vectors based on API call information and classifying four types of malicious samples. Ivan et al. [32] conducted a comparative analysis on a sample library of 470 samples using KNN, Naive Bayes, SVM, J48, and MLP. However, the reliability of their approach is challenging to ensure.

The aforementioned methodologies entail manual analysis for feature vector extraction, employing traditional machine learning algorithms for classification. In this paper, we employ the Convolutional Neural Network (CNN) algorithm to detect malicious dynamic behavior in samples, drawing inspiration from CNN applications in image processing. A notable feature of this method lies in its elimination of the need for manual feature vector extraction. Instead, the algorithm autonomously learns specific features based on the inherent dynamic behavior information within the samples.

The method proposed in this paper involves training a deep learning model using samples from known classes to detect new and unidentified malware during testing. Convolutional Neural Networks (CNNs) exhibit advantages over alternative algorithms in image classification. In general, the fundamental structure of a CNN comprises two layers, the first being the feature extraction layer. Each neuron's input is connected to the local receptive field of the preceding layer to extract local features. Following the extraction of local features, their positional relationships with other features are determined. The second layer is the feature mapping layer, characterized by multiple feature maps within each network computation layer. Each feature map can be visualized as a plane, where all neurons within the plane share identical weights. The feature map structure incorporates the sigmoid function with a compact influence function kernel serving as the activation function, promoting displacement invariance. The sharing of weights among neurons on the mapping surface effectively minimizes the number of free parameters in the network. This study utilizes three convolution kernels of varying lengths (3, 4, 5). The width of the convolution kernel matches the length of the word vector. Following the convolution operation, the original grayscale image is transformed into a column vector, akin to the N-Gram algorithm. The use of a convolution kernel of length 3 extracts features from three adjacent dynamic behaviors. Multiple convolution kernels are applied at each scale, with two convolution kernels per scale. A total of 128 convolution kernels are employed in this paper. Subsequent to convolution, max pooling is applied to each convolution result, yielding the maximum value in the column vector. Each column vector is reduced to a 1x1 value. The maximum values corresponding to all convolution kernel results are concatenated to form a fully connected layer. The third step involves utilizing the softmax function for binary classification processing. The constructed CNN model is shown in Fig.8.

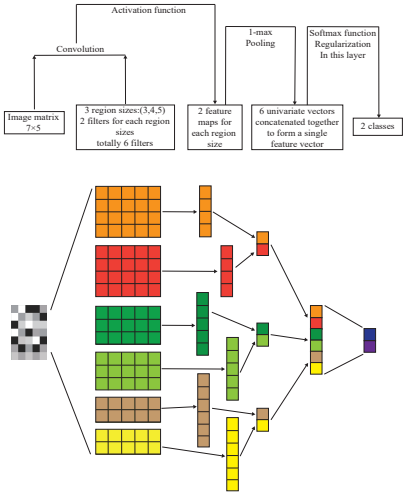

Fig. 8: CNN model

V. THE CONSTRUCTION OF MULTI-DIMENSIONAL DYNAMIC BEHAVIOR IMAGE DATASET

There are usually many problems in obtaining malicious samples. Malicious samples shared by public websites such as VirusShare [33], VirusTotal are protected by copyright law and are prohibited from being shared. For inexperienced malware related workers, there are also security risks for them that do not take security measures to save malicious samples. Considering of the legality of sharing malicious samples and the security of saving malicious samples, this paper chooses Malwarebazaar sample library as the source of malicious samples. Malwarebazaar encrypts and packages all malicious samples into a compressed package named hash value; after downloading and obtaining the corresponding password, the compressed package can be decompressed. This operation ensures the uniqueness of the malicious samples and the security of acquisition. Malwarebazaar sample library uploads newly discovered types of malicious samples every day, making the sample library rich in sample types and real-time. Malwarebazaar also collects and packages daily and weekly malicious samples, and provides calling APIs interface, which improves the convenience of malware-related workers using data. In addition, obtaining malicious samples online usually requires complex registration and review processes, but Malwarebazaar eliminates this step and has no restrictions on downloads for all users.

We compare the commonly used malware visualization dataset with the self-built dataset, as shown in Table I.

Sample selection forms the foundational data for model training and testing. While public datasets are easy to access, concerns related to privacy leakage may arise due to various feature extraction methods. Additionally, the large volume of data in public datasets, coupled with a mix of data types, poses challenges in effective filtering. Considering the inherent

TABLE I: Public Malware Datasets Comparison

Name	Size(GB)	Feature Type
Malimg	1.09	Static
Ember	9.38	Static
MalwVis(300*300)	2.91	Static
Our dataset	0.53	Dynamic

issues with public datasets, this paper opts to design and construct a behavioral dataset.

The construction process is as follows: In the first step, 1559 malware samples were downloaded from the MalwareBazaar sample library, and 1150 benign software samples were collected from other open-source platforms to manually construct the sample library. In the subsequent step, these samples were subjected to Cuckoo Sandbox for execution, and dynamic behavior reports generated within Cuckoo Sandbox were collected. The third step involved filtering the report files in the behavioral dataset, removing those that did not meet the established standards. The original report files were employed for the construction of the behavioral dataset. As illustrated in Fig.9.

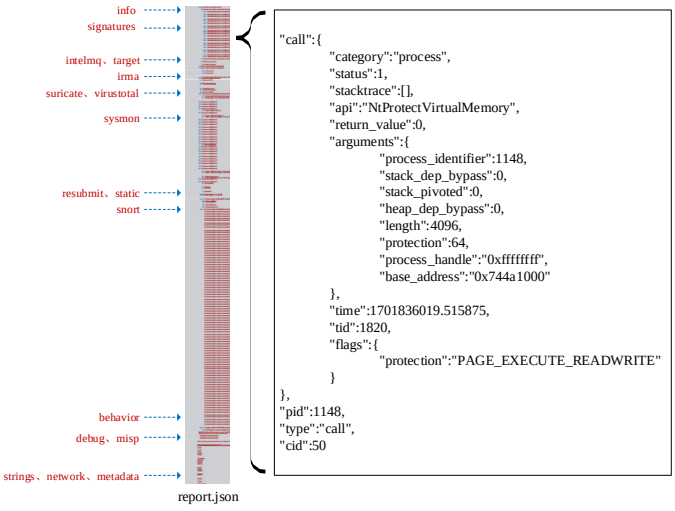

Fig. 9: Original Report Example

Cuckoo Sandbox is an automated malware analysis system that can detect malware behavior and generate dynamic behavior reports under isolation conditions, ensuring safe execution and independent analysis of malicious code samples. The structure of Cuckoo Sandbox is shown in Fig.10.

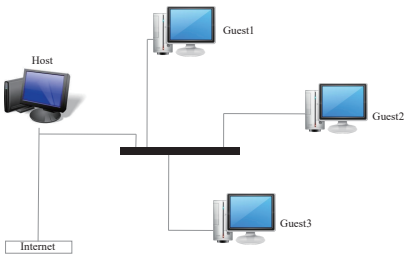

Fig. 10: Cuckoo Sandbox Structure

The criteria for selecting reports are delineated as follows:

Diversity: Reports encompassing a broad spectrum of categories are chosen. It is imperative for behavioral datasets to exhibit diversity across various types of malware, thereby enhancing the demonstration of the efficacy of detection endeavors.

Complexity: The chosen reports should exhibit complexity and involve multiple steps. Within the sample library, certain malicious samples feature simplistic attack steps and lack substantial contextual information. Such reports hold little significance for subsequent analysis. Ultimately, 848 malware reports and 619 non-malware reports were selected to construct the behavioral dataset.

VI. EXPERIMENTAL RESULTS AND EVALUATIONS

In this section, we first introduce experimental hardware and software. Then, we make comparative experiments. The detialed process is as follow.

A. Experimental Environment

The hardware setup comprises a PC equipped with an Intel® Core™ i7-9750H CPU, a GeForce RTX 2080 GPU, and 16GB RAM. The experimental sandbox environment is configured as follows: the virtual machine runs on Ubuntu 16.04 LTS, while the sandbox operating system is Windows 7 Professional. Common software is installed, and for data processing and model construction, PyCharm is utilized. The programming language employed is Python, and the primary operating system is Windows 10. The training set and test set data split is maintained at a ratio of 8 to 2.

B. Performance Comparison

To analyze our scheme better, we conduct comparative experiments against other malware detection methods. In this experiment, we employ the comparative experimental method of sample gradient increment to assess the performance of Long Short-Term Memory (LSTM), CNN-LSTM [34], and our previous work using the self-constructed sample library. LSTM represents a neural network architecture utilizing fully connected layers for comparison. CNN-LSTM is a deep learning method based on LSTM. Our prior work [35] proposed a visual malware detection model and a concrete scheme based on multi-dimensional dynamic behaviors, encompassing API and network operation information. The comparison results are illustrated in Fig.11.

As illustrated in Fig.11, the detection accuracy of the four methods exhibits fluctuations with various sample numbers. Both LSTM and CNN-LSTM display a rising trend after the sample size surpasses 200 and 400. Our previous work performs relatively stable. Conversely, the accuracy of the proposed solution hovers around 97.5% after the sample size exceeds 200. Throughout the experiment, it was observed that, when the sample size is below 600, factors such as the richness and relevance of contextual information may lead to significant fluctuations in the accuracy rate. To mitigate the impact of sample variations on the experiment, this study conducted multiple random sample selections in the early stage of small

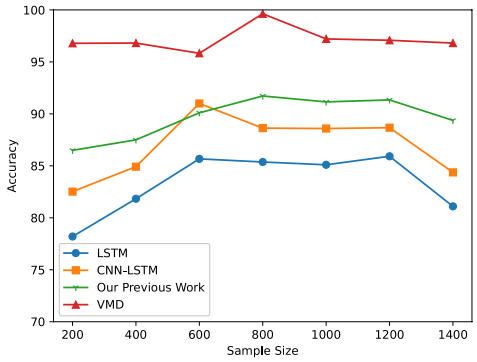

Fig. 11: Four Methods Accuracy

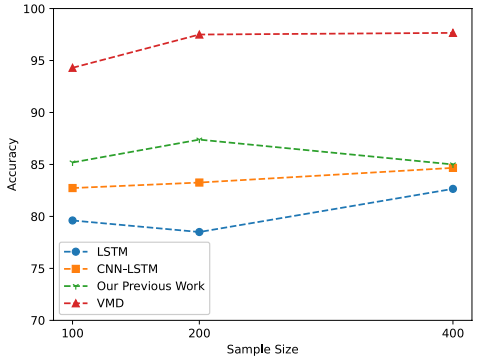

Fig. 12: Four Methods Accuracy with Sample Size Smaller than 600

samples. The average accuracy from multiple random repeated tests under small sample conditions is shown in Fig.12.

As depicted in Fig.12, it is evident that, when the sample size is below 600, the proposed solution in this paper generally outperforms the three compared methods. With further expansion of the sample size, the intrinsic properties of the sample cease to be the primary factor affecting experimental performance. The subsequent table presents the experimental results of the four methods on the self-constructed sample library.

As observed in Table II, the accuracy, precision, and recall rate of the proposed method surpass those of the three compared methods. Through experimental analysis, the overall superiority of the solution presented in this paper may be attributed to the chosen text preprocessing approach. Unlike other detection methods that rely on the feature representation of input data, the good performance of this paper's solution is contingent on a robust text preprocessing solution. Inadequate feature extraction processes or inappropriate feature selection may lead to a less accurate model in detecting malware. The subsequent table illustrates the performance of the four methods across different sample sizes.

Precision denotes the ratio of true positive samples to all detected positive samples, while recall signifies the proportion of detected positives among all potential positive samples. These two metrics provide insights into the model's performance from different perspectives. As depicted in Table III, when the sample size is below 400, the accuracy of this method

TABLE II: Experiment Results

Method	Accuracy	Precision	Recall
LSTM	81.08%	81.16%	80.11%
CNN-LSTM	84.37%	84.51%	86.32%
Our Previous Work	89.75%	98.08%	84.30%
VMD	96.81%	95.93%	97.17%

is lower than that of CNN-LSTM. Additionally, when the sample size is below 200, the accuracy of this method is lower than that of LSTM. While the overall trend of our previous method is relatively stable, its accuracy, precision, and recall are inferior to the method proposed in this paper. However, the recall rate of this method generally outperforms the other two methods in the experiment. This experiment also investigates the performance of the four methods in detecting unknown malware. The behavioral dataset is partitioned into a training set and a test set in a 4:1 ratio. The experimental results are presented in the following table.

TABLE III: Experimental Data of Four Methods at Different Sample Size

Method	Samples	Accuracy	Precision	Recall
LSTM	100	79.73%	82.36%	84.83%
	200	78.20%	81.10%	79.80%
	400	81.83%	79.59%	81.35%
	600	85.67%	86.31%	85.67%
	800	85.37%	85.92%	84.68%
	1000	85.10%	85.23%	85.11%
	1200	85.92%	86.16%	80.18%
LSTM-CNN	100	82.04%	81.02%	81.08%
	200	82.52%	82.87%	84.00%
	400	84.92%	83.77%	87.63%
	600	91.00%	91.76%	90.97%
	800	88.63%	89.00%	88.61%
	1000	88.59%	88.80%	86.72%
	1200	87.67%	87.86%	86.32%
Previous work	100	88.89%	83.33%	90.91%
	200	86.49%	93.33%	87.50%
	400	87.50%	92.75%	74.36%
	600	91.73%	97.96%	81.36%
	800	91.72%	92.50%	91.36%
	1000	91.15%	95.83%	83.13%
	1200	91.34%	98.06%	84.67%
VMD	100	94.00%	89.21%	100.00%
	200	96.79%	95.33%	99.00%
	400	96.81%	96.44%	98.19%
	600	95.83%	99.28%	92.33%
	800	99.63%	99.50%	99.75%
	1000	97.21%	94.87%	99.80%
	1200	97.08%	94.63%	99.83%

As evident from Table IV, the accuracy and recall rate of this method surpass those of the other three methods, although the accuracy exhibits a decline. Upon analysis of the accuracy decline, it may be attributed to the similarity in the types of APIs called by some benign and malicious software when the dataset is small, leading to model misjudgments. Nevertheless,

the overall performance of this method tends to outperform the three compared methods.

TABLE IV: Unknown Malware Test Result

Method	Accuracy	Precision	Recall
LSTM	79.10%	79.59%	79.21%
CNN-LSTM	80.77%	81.27%	80.81%
Our Previous Work	83.90%	84.92%	88.43%
VMD	84.45%	79.45%	98.05%

VII. CONCLUSION

In this paper, we constructed a visual malware detection model that integrates multi-dimensional dynamic behavioral information with deep learning. First, we processed our samples dynamic behavior as grayscale images, used image feature descriptor to characterize malware. Then, we trained our models to classify malware and goodware. However, in the current work, we only focus on binary classification. It should consider expanding the number of label assortments and samples to facilitate multi-classification experiments in the future work.

ACKNOWLEDGEMENTS

This work was supported by the National Key R&D Program of China(Grant No. 2023YFB3107500), the Major Research plan of the National Natural Science Foundation of China (Grant No. 92267204), the Key R&D Program of Shaanxi Province (2021ZDLGY03-10), Shandong Provincial Natural Science Foundation (ZR2021LZH006), the Chinese Postdoctoral Science Foundation (2021MD703967).

REFERENCES

[1] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21 235–21 245, 2019.

[2] M. F. Zolkipli and A. Jantan, "A framework for malware detection using combination technique and signature generation," in *2010 Second International Conference on Computer Research and Development*. IEEE, 2010, pp. 196–199.

[3] H. R. Borojerdi and M. Abadi, "Malhunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection," in *ICCKE 2013*. IEEE, 2013, pp. 430–436.

[4] M. Zheng, M. Sun, and J. C. Lui, "Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013, pp. 163–171.

[5] G. Wagener, R. State, and A. Dulaunoy, "Malware behaviour analysis," *Journal in Computer Virology*, vol. 4, pp. 279–287, 2008.

[6] Y. Fukushima, A. Sakai, Y. Hori, and K. Sakurai, "A behavior based malware detection scheme for avoiding false positive," in *2010 6th IEEE Workshop on Secure Network Protocols*. IEEE, 2010, pp. 79–84.

[7] M. Chandramohan, H. B. K. Tan, L. C. Briand, L. K. Shar, and B. M. Padmanabhuni, "A scalable approach for malware detection through bounded feature space behavior modeling," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2013, pp. 312–322.

[8] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, 2015.

[9] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "Cimds: adapting postprocessing techniques of associative classification for malware detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 3, pp. 298–307, 2010.

[10] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*. IEEE, 2013, pp. 113–120.

[11] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3422–3426.

[12] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in android malware detection," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014, pp. 371–372.

[13] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, 2018, pp. 81–85.

[14] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 11–20.

[15] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "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.

[16] W. Huang and J. W. Stokes, "Mtnet: a multi-task neural network for dynamic malware classification," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*. Springer, 2016, pp. 399–418.

[17] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv Preprint ArXiv:1409.1556*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[23] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, 2011, pp. 1–7.

[24] H. Naeem, B. Guo, M. R. Naeem, F. Ullah, H. Aldabbas, and M. S. Javed, "Identification of malicious code variants based on image visualization," *Computers & Electrical Engineering*, vol. 76, pp. 225–237, 2019.

[25] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of iot malware based on image recognition," in *2018 IEEE 42Nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2018, pp. 664–669.

[26] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, pp. 1–14, 2015.

[27] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *ArXiv Preprint ArXiv:1804.04637*, 2018.

[28] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, "Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision," *Computers & Security*, vol. 103, p. 102166, 2021.

[29] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on cnns and multi-objective algorithm," *Journal of Parallel and Distributed Computing*, vol. 129, pp. 50–58, 2019.

[30] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[31] R. S. Pirscoveanu, S. S. Hansen, T. M. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech, "Analysis of malware behavior: Type classification using machine learning," in *2015 International Conference on*

Cyber Situational Awareness, Data Analytics and Assessment (CyberSA). IEEE, 2015, pp. 1–7.
[32] I. Firdausi, A. Erwin, A. S. Nugroho *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*. IEEE, 2010, pp. 201–203.
[33] C. Forensics, "Because sharing is caring," https://virusshare.com/.
[34] F. O. Catak, A. F. Yazı, O. Elezaj, and J. Ahmed, "Deep learning based sequential model for malware analysis using windows exe api calls," *PeerJ Computer Science*, vol. 6, p. e285, Jul. 2020. [Online]. Available: https://doi.org/10.7717/peerj-cs.285
[35] Z. Ma, Z. Zhang, C. Liu, T. Hu, H. Li, and B. Ren, "Visualizable malware detection based on multi-dimension dynamic behaviors," in *2022 International Conference on Networking and Network Applications (NaNA)*. IEEE, 2022, pp. 247–252.

YuLong Cui received the B.S. degree in Software Engineering from Henan University in 2022. He is currently a M.D. student in the School of Computer Science and Technology, Xidian University. His research interests include system security for intelligent swarm.

ZhiWei Zhang received the Ph.D. degree in cryptography from Xidian University, Xi'an, Shaanxi, China, in 2019. His research interest includes authentication, access control, data storage security in cloud computing and endogenous security in intelligent swarms.

ZhiDong Ma is currently a M.D. student in the School of Computer Science and Technology, Xidian University. His research interests include wireless network security and intelligent system endogenous security.

ZeHan Chen received the B.S. degree in Software Engineering from Guangdong University of Finance and Economics in 2020. He is currently a Ph.D. student in the School of Computer Science and Technology, Xidian University. His research interests include task offloading and system security for intelligent swarm.

YuZi Wang received the B.Eng. degree in computer science and technology from Xidian University, Xi' an, shanxi, China, in 2023. His research interest includes deep learning, artificial intelligence, network security, Anomaly Detection.

YuLong Shen received the B.S. and M.S. degrees in computer science and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively. He is currently a Professor with the School of Computer Science and Technology, Xidian University. His research interests include wireless network security and cloud computing security.