Digital Investigation 27 (2018) 30–37

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

A malware classification method based on memory dump grayscale image

Yusheng Dai [a], Hui Li [a], Yekui Qian [b, *], Xidong Lu [a]

[a] School of Electronics and Information, Northwestern Polytechnical University, China
[b] 24 Jianshedong Road, Zhengzhou, Henan, 450052, China

ARTICLE INFO

Article history:
Received 28 May 2018
Received in revised form
17 September 2018
Accepted 20 September 2018
Available online 25 September 2018

Keywords:
Dynamic analysis
Hardware features
Memory dump
Malware classification

ABSTRACT

Effective analysis of malware is of great significance in guaranteeing the reliability of the system operation. Malware can easily escape from existing dynamic analysis methods. Aiming at the deficiencies of current methods for detecting malware dynamically, a method of using hardware features is proposed, namely, a memory dump file is extracted and converted into a grayscale image, the image is converted into a fixed size, and the image feature is extracted using histogram of gradient, and the currently popular classifier algorithm is used to classify malware. Experiments are conducted using actual malware samples and the effectiveness of using memory dump file image is verified. This method is superior to the recently proposed hardware performance counter detection method.

Introduction

In nowadays, with the rapid development of information technology, governments and financial networks of various countries have been frequently attacked by malware, which seriously threatens the finance, national defense, and education industries of the countries. According to the Kaspersky report (Kaspersky lab, 2016), the number of daily generated new malware in 2017 reached 360,000, an increase of 11.7% on a year-on-year basis. A joint report published by McAfee and CISCO (New global cybersecurity, 2018) stated that the losses caused by cybercrime last year reached USD 60 billion, equivalent to 0.8% of global GDP. Since cybercrime usually requires the use of specific malware to invade and attack the target, it is of great significance to be able to effectively analysis malware.

Today's Anti-Virus software (AV) often use a signature-based detection method (Ramzan et al.,). However, when a new variant of malware occurs, the AV cannot get a new malware signature in time, so this method cannot effectively detect new malware. In the research field, malware analysis technology is usually divided into two major categories: static analysis and dynamic analysis (Idika and Mathur). Static analysis can quickly identify malware, but it is vulnerable to obfuscation attacks such as encryption and packaging. Dynamic analysis can effectively overcome the shortcomings of static analysis technology, but it is vulnerable to malware evasion.

Since the AV software itself is a kind of software, the defect density of the detection software itself is about the same with that of the normal software (Tang et al., 2014). Therefore, a considerable amount of research has recently focused on the analysis of malware based on hardware features. The use of hardware features can avoid the deficiencies of software features to solve the malware evasion problem in dynamic analysis. Demme et al. (2013) proved that using performance counters (e.g., IPC, cache behavior, memory behavior, etc.) can effectively classify malware by extracting hardware features. Based on above research, Tang et al. (2014) used hardware performance counters (HPC) combined with unsupervised methods to detect malware. However, the above method cannot fully describe the behavior of malware with HPC and the classification accuracy is insufficient.

For above problems, this paper proposes a malware dynamic analysis method based on memory dump data, which converts malware binary files into grayscale images, uses the texture features of the image and the classification method of multilayer perceptron (MLP) to classify the images. During the running of malware, the data stored in the memory has enough information to determine whether it is malicious through detection, the memory data is extracted to generate a dump file and converted to a

* Corresponding author.
E-mail address: qyk1129@163.com (Y. Qian).

https://doi.org/10.1016/j.diin.2018.09.006
1742-2876/

grayscale image, and an image scaling algorithm is used to convert the image to a uniform size and classifiers are used to classify malware.

This paper mainly contributes in 3 points as follows:

1. We propose a malware extraction method based on memory dump. The sandbox is used to monitor the malware process memory, and the memory data dump file is converted into a grayscale image, and the feature is extracted using a histogram of gradient (HOG). To certain extent, this method can solve the malware evasion problem of dynamic analysis.
2. We propose a method for converting into grayscale image by using memory dump data. In combination with a multilayer perceptron classifier, malware can be effectively classified.
3. The effectiveness of the proposed method is verified by using the actual backdoor malware dataset for actual classification and conducting experimental evaluation.

The remainder of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces memory dump and presents a malware classification method that uses memory dump. Section 4 presents the evaluation of our experiments and discussed the results. Finally, Section 5 concludes the paper.

Related work

With the rapid development of information technology today, the field of malware analysis has attracted extensive attention from both academic and business communities. Malware analysis is generally classified into static analysis and dynamic analysis. Early static analysis methods use signature-based detection technology (Ramzan et al.,). This method is applicable to the commercial field, but it cannot effectively detect malware variants and emerging malware. In the current advanced static analysis technology, Arp et al. (2014) used feature vectors based on a combination of semantics and grammar to detect Android malware. Feng et al. (Feng et al.,) proposed to combine automatic learning semantics with the largest suspicious common subgraph to speculate unknown malware signature technology, but these methods are subject to static analysis defects. Nataraj et al. (Nataraj et al., 2011a, 2011b; Nataraj and Manjunath, 2016) proposed a method of converting a malware binary file into a grayscale image for the first time, and used the texture features of the image and the Nearest Neighbor (k-NN) classifier to classify the images. The use of a visual method makes up for the defect that the static analysis is easily to be puzzled by methods such as confusion and encryption, but it still cannot solve the problem of code filling.

Dynamic analysis can effectively overcome the shortcomings of static analysis. It can monitor the behavior of the program when it is running (Egele et al., 2008) to detect whether the program is malicious. However, the dynamic analysis also has some disadvantages. The dynamic analysis of malware cannot effectively traverse all possible paths that the code may execute. At present, a considerable portion of malware can check its environment, and it will automatically crash or shut down when it is found in a virtual environment. Kirat et al. (Kirat et al., 2011; Kirat et al.,) proposed that the bare-metal system could avoid the problem that the malware could detect the environment. However, the environment using the bare-metal system requires a complete network architecture and a dedicated host to collect machines running results, and bare-metal machines running results can only be recorded at the startup and shutdown. In addition, some studies (Liu et al., 2013; Andronio et al., 2015; Smutz and Stavrou, 2016) use a combination of static analysis and dynamic analysis and hope to eliminate the disadvantages including static analysis susceptibility to confusion and dynamic analysis susceptibility to escaping at the same time. However, none of these works effectively targets the defects of the software itself.

Malware analysis method based on hardware features has received more attention and been researched in recent years (Demme et al., 2013; Tang et al., 2014; Ozsoy et al., 2015, 2016; Khasawneh et al., 2015). It uses a new feature method, as the vulnerability of detection software and general software is not essentially different and both of them are vulnerable to attack. In the process of dynamic analysis, the simulation environment is easily detected by malware, resulting in evade. Demme et al. (2013) found that collecting performance counter information of malware at runtime effectively distinguished malware through offline analysis and verified the effectiveness of using hardware performance counter (HPC). By using HPC and its related events, Tang et al. (2014) demonstrated that using off-line unsupervised analysis method could also effectively analyze new types of malware or family evolution. Ozsoy et al., 2015, 2016 and Khasawneh et al. (2015) used a two-level detection method and a variety of low-level hardware features to perform pre-detection, increased the weight of suspicious files, and used software detectors for re-examination. However, these hardware features do not fully describe the program behavior, so further improvement in the accuracy of the analysis is required. Here are some methods using memory dump analysis: Korkin et al. (Korkin and Nesterov, 2015) obtained evidence in the memory dump file to detect Rootkit code; Case et al. (Case and Iii, 2016) used memory dump to detect the object-C malware; Javaheri et al. (Javaheri and Hosseinzadeh, 2017) proposed to use different time slices of dump file to extract the important data structure, and classify the dump data into different kinds of malware according to the data in the dump; however, these analysis methods of using memory dump data are still based on heuristic analysis.

In this paper, according to method based on converting the memory dump file binary data into grayscale image, the true malware dataset is used to extract the dump data in the sandbox running environment, and the use of memory dump data can effectively resist malware detection environment and adopt escape means. The conversion of binary data to grayscale image file can make full use of the similarity of malware in memory behavior, and can reduce the complexity of the analysis algorithm. In the experiment, the effectiveness of this method on malware classification has been proved to be better than that of partial recent researches.

Dump classification method

The malware classification method in this paper is mainly divided into four steps: the first step is to use the sandbox to extract malware memory dump file at the runtime; the second step is to map the malware memory dump file into grayscale image; the third step is to conduct feature extraction from the nondestructive grayscale image obtained from the second step, compress based on the bicubic interpolation method commonly used in image processing and extract the feature vectors; the last step is malware classification through the machine learning classification method. The main process of the experiment is shown in Fig. 1.

Memory dump

In order to detect the malware effectively at runtime, we are aware that no matter what program it is, the data in memory will be inevitably exposed under supervision. Memory dump refers to the volatile data extracted from the computer physical memory and virtual memory, and it is the important data used in this computer branch science of memory forensics. Memory dump usually has full

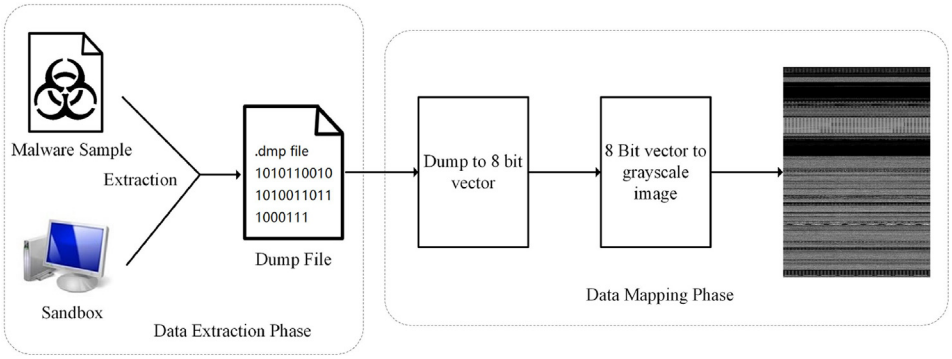

Fig. 1. Classification process of malware memory dump.

memory dump, core dump, process dump, and so on. Here we extract the process memory dump file to analyze the process to determine whether the process is malicious.

In the operating system, the system is responsible for memory management, which can be summarized into three mechanisms: virtual address space management; physical page management; address translation and page swapping. All processes run in their own virtual address space, and each process has its own private memory space. The memory dump file analyzed in this paper, namely full memory data of single process, includes the data in the actual physical memory space and the virtual memory space data on the disk. Under normal conditions, the layout of each process in the operating system is shown in Fig. 2.

The process memory usually contains the following:

- Dynamic link library (DLLs): The DLL file that this storage process needs to call.
- Environment variables: This memory has stored the process environment variables, such as its executable path, temporary directory, home folder, etc.

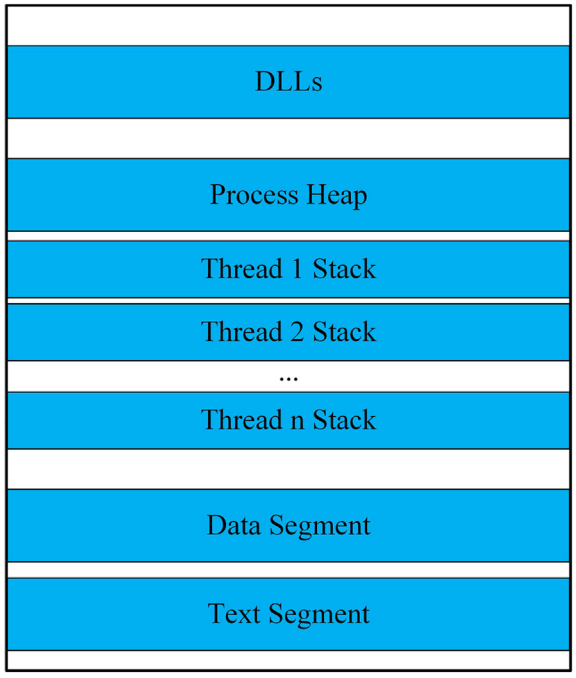

Fig. 2. Typical process memory layout.

- Process heap: It is an area of memory allocation of dynamic process at runtime and is the area to save all the memory of the process dynamically applied at present as well as the data in the memory, such as strings dynamically allocated in the process, buffer data of network socket.
- Thread stack: The system has provided each thread of each process with stack memory, which contains function parameters, function return address, and local variables.
- Data segment: It includes the BSS (Block Start by Symbol) segment of uninitialized data and initialized data, and this segment of data contains global variables, constants, etc. used in the program.
- Text segment: It contains the main code instructions and read/write variables of the application program, and it is read-only in memory to prevent the process from accidentally modifying its own instructions through the error pointer. Fig. 3 has shown a part of text segment of process dump contents.

The layout of data in the process memory is usually based on the layout of above blocks from high address to low address; however, the OS usually uses Address Space Layout Randomized (ASLR), which means that these blocks are not often continuous, especially in the smaller memory environment, the distribution of each block is more random (Korkin and Nesterov, 2015). Hence, in the experiment, we set up a large memory ($\geq$2GB) and used the extraction tool of the memory dump to extract the dump file from the contiguous memory address space.

Dump extraction

In order to extract the malware memory dump file, we used the cuckoo sandbox (Cuckoo sandbox) to dynamically analyze malware. The sandbox is a virtual machine environment allowing us to monitor the running behavior (e.g. registry modification, disk reading, API call sequence, etc.) results of the program running in its environment, and determine whether the current running program is malicious based on the analysis of results.

We built the Procdump program (Procdump) in cuckoo sandbox. We use the ma command to extract the dump of the monitored process. The dump file extracted by the ma command includes the entire virtual address space of the process, regardless of whether the memory is paged in to RAM. To prevent malware using evasion techniques, we monitor the malware in the guest machine. The Procdump program create a dump file as the final result before the sandbox execution timeout or when malware crashes. Then we save the dump file to the host machine, the extraction process is shown as the data extraction phase in Fig. 1, and the extracted dump file is used for us to convert the file into grayscale image and

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
002a3590	4d	61	6e	61	67	65	72	00	4e	74	52	65	70	6c	61	63	Manager.NtReplac
002a35a0	65	4b	65	79	00	4e	74	52	65	70	6c	61	63	65	50	61	eKey.NtReplacePa
002a35b0	72	74	69	74	69	6f	6e	55	6e	69	74	00	4e	74	52	65	rtitionUnit.NtRe
002a35c0	70	6c	79	50	6f	72	74	00	4e	74	52	65	70	6c	79	57	plyPort.NtReplyW
002a35d0	61	69	74	52	65	63	65	69	76	65	50	6f	72	74	00	4e	aitReceivePort.N
002a35e0	74	52	65	70	6c	79	57	61	69	74	52	65	63	65	69	76	tReplyWaitReceiv
002a35f0	65	50	6f	72	74	45	78	00	4e	74	52	65	70	6c	79	57	ePortEx.NtReplyW
002a3600	61	69	74	52	65	70	6c	79	50	6f	72	74	00	4e	74	52	aitReplyPort.NtR

Fig. 3. Part of API sequence stored in text segment.

retain for further research.

Dump file visualization

In order to ensure that the extracted dump file can be identified and analyzed by machine learning algorithm, here the extracted binary dump file is mapped into grayscale image. For a given binary file, read 8bit unsigned integer vector from the starting position in the file, set the array line width to a constant value, the height changes with the size of the dump file, then the entire dump file will generate a two-dimensional array. The value range of each element in this matrix is [0, 255] (0 is black, 255 is white). According to the observation experience (Nataraj et al., 2011b) and the size of the extracted memory data in the experiment (the dump file is usually bigger than 3MB), the file size greater than 4MB is recommended to use 4096 pixel width; the file size less than 4MB should use 2048 pixel width.

Fig. 4 has shown the backdoor malware of two families, which usually bypass the security control of OS to infiltrate or destroy the victims system. Though the backdoor malware has many specific behaviors, there is also a similarity in the behavior embodied in the memory of the same family. Fig. 4 shows that grayscale image (a) is Bifrose family, and (b) is Ceckno family.

Image feature extraction

As the computer cannot directly recognize the image, it is need to extract the features of the image for vector representation. In order to extract feature vectors with equal numerical values, and to ensure that the data in feature vectors can effectively save features in the image, bicubic interpolation is used to scale the grayscale image converted by malware, and then HOG (Dalal and Triggs, 2005) is used for feature extraction of the image.

Bicubic interpolation

As the operation behaviors of every malware process in the sandbox environment are different, the dump data of malicious process extracted are inconsistent, in order to extract the feature vectors with the same dimension, and under the condition that the images are not obviously distorted, the bicubic interpolation can compress the images to the same pixel size. In image processing, the bicubic interpolation can consider 4*4 pixels, this algorithm can effectively compromise with the grayscale effect of adjacent points within the area and the effect of grayscale change rate of every adjacent point. Bicubic interpolation can be expressed in the following equation:

$$p(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{1}$$

Wherein, $p(x,y)$ are continuous with continuous derivatives within the unit area, then they are spliced together through these bicubic curved surfaces, so as to guarantee that the derivatives and the boundaries are matched.

Histogram of gradient

The feature of Histogram of Gradient is a kind of feature descriptor used for detecting objects in computer image processing, a feature is constructed through calculating and counting the HOG in local areas. In our previous work, using this method can

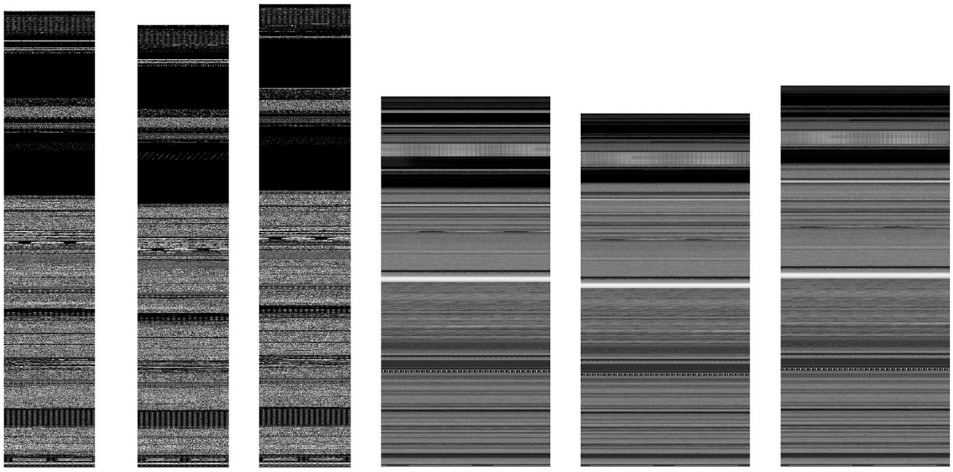

Fig. 4. Grayscale image of malware memory dump in different families.

effectively extract the features of grayscale image to further classify the malwares (Lu et al.,).

The extraction of HOG features firstly needs to calculate the gradient value, use $[-1,0,1]$ sub gradient template for the horizontal direction x of (x,y) point, use $[-1,0,1]^T$ sub gradient template for the vertical direction y, respectively conduct convolution calculation for the two directions, and calculate the total gradient magnitude and direction of image:

$$Grad(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{2}$$

$$\alpha(x,y) = \tan^{-1}\frac{G_x(x,y)}{G_y(x,y)} \tag{3}$$

In Equation (2), $Grad(x,y)$ is the gradient magnitude, in Equation (3), (x,y) is the gradient direction, wherein, it is respectively the gradient component in direction x and gradient component in direction y.

Then normalize the contrast of the cell unit, in HOG, the scale for local contrast statistics is one block, the size of block is assumed as 16*16, wherein, it includes 2*2 cells, the moving step length of window is 8 pixels, thus, there is superposition between blocks, so as to guarantee that different normalized results of the same cell can make contributions to the final HOG vector. In the normalization, the feature vectors of every cell are cascaded based on the voting results, the vector feature of block is obtained, adopt L2-norm for the vector feature normalization of block:

$$v' = \frac{v}{\sqrt{\|v\|_2 + \varepsilon}} \tag{4}$$

In Equation (4), $\|v\|_2$ is the L2 norm of feature vector v, the L2 norm, ε is a constant that is not zero, the HOG feature vector finally obtained is $f = \{v'_1, v'_2, v'_3, \ldots, v'_n\}$.

Malware classification

In 3.4, we have extracted the HOG feature vector from the grayscale image with the fixed size, this section will take the HOG feature vector of grayscale image of memory dump data as the input, adopt multilayer perceptron as the classifier, realizing the malware classification.

Input layer

This layer is used to receive the HOG feature vector, set the number of nerve cells in the input layer of the neural network model as the feature dimension number extracted by the grayscale image.

Hidden layer

It is constituted by the vector $h(x)$. For the weight matrix $W^{(1)} \in R^{D \times Dh}$, the input layer is completely connected to the hidden layer, activating the Rectified Linear Unit (ReLU), wherein, the hidden vector function is expressed as:

$$h(x) = Re\left(b^{(1)} + W^{(1)}x\right) \tag{5}$$

For the mapping function $f = R^D \to R^L$ from one layer to the next layer, D is the magnitude of the input vector x, L is the magnitude of the output vector $f(x)$, the conversion formula is as follows:

$$f(x) = G\left(b^{(k+1)} + W^{(k+1)}\left(Re\left(b^{(k)} + W^{(k)x}\right)\right)\right) \tag{6}$$

Wherein, $b^{(k)}$ and $b^{(k+1)}$ represent the offset vector of two adjacent layers, $W^{(k)}$ and $W^{(k+1)}$ are the weight matrix of two adjacent layers activating the functions of G, Re. G is the softmax function used to convert the results of neural network forward propagation into the probability distribution.

Output layer

This layer is the end of the entire neural network with the number equal to those of training malware types. We set a MLP of 4 hidden layers and one input layer, then the output layer vector is represented as:

$$o(x) = G\left(b^{(5)} + W^{(5)}h(x)\right) \tag{7}$$

In order to obtain the optimal multilayer perceptron (MLP) model, we need to train the loss function of network to make output layer vector get close to the output form of one-hot so as to predict the categories of malware. At this time, the weight and bias of the hidden layer need to be modified. In this paper, cross entropy is used as the loss function to measure the deviation of real results and model prediction, with setting number of categories as k, the error of prediction probability distribution $\widehat{x}_i$ and real probability distribution x_i is represented by using cross entropy equation as:

$$C(\theta) = -\sum_{i=i}^{k} x_i \log \widehat{x}_i \tag{8}$$

According to the deviation result obtained from Equation (8), stochastic gradient descent algorithm (SGD) (Ketkar) is used to fine-tune the bias and weight parameters of all neurons. Therefore, we can obtain the training algorithm and steps of the multilayer perceptron (MLP) model as follows:

(a) Initialize the connection weights and bias in network (it is determined by the random seed), in following equation, s stands for the node number of hidden layer.

$$(W,b) = initp(X,s) \tag{9}$$

(b) For k output vectors with classified objects as 1 or 0, if the corresponding output vector is one-hot form, the iteration will be terminated. Otherwise, the weight and bias value shall be corrected according to the reverse direction of the random gradient.
(c) Repeat step (b) until the network output is consistent with the training objective.

Experimental evaluation

Experimental environment and experimental data

The computer environment for the running of sandbox environment is CPU using CPU of Intel(R) Core(TM) i5-6500 @3.20GHz; using 8GB DDR3 RAM. The Host machine of cuckoo sandbox is installed under Ubuntu 16.04 system environment with Guest machine using Windows 7 x86 system, 2GB/512MB RAM.

The experimental classifier algorithm runs on a workstation and the hardware environment CPU uses Intel Core (TM) i7-6800K; 32GB dual channel DDR4 RAM.

The malware data used in this paper comes from Open Malware Benchmark (Openmalwarebenchmark, 2018), with authorized using a total of 1984 malware and five family categories, collected between 2012 and 2015. Determining the family through VirusTotal (Virustotal), after screening, the basic information of final available malware and family is shown in Table 1.

Table 1
Malware data set.

Malware family	Category No.	Quantity	Example MD5
Agent	0	125	6c2069702e64f18b2843ee97e86fd08d
Bifrose	1	345	6c8a7b9758ce895e3ec6092c18081580
Blackhole	2	151	13906517ab38d95de465fc591a4c7ae0
Ceckno	3	262	2c5100be3a4d6c8d9f8c06d54106f2d2
Ciadoor	4	55	53717a910fcf3a88ed0e296faa94d317
Hupigon	5	178	15964339b2cade192f89bcad6688619c
Virut	6	344	b36b24cae0973c3b374a34feb8a3b013
PowerShell	7	8	affb1af048a4f249f80cbf701711d1e2

Assessment criterion

This experiment adopts Accuracy rate (Acc), Precision rate (P), Recall rate (R) and F1-Score totally four indicators to measure the effect of classification which is performed by the method in this paper combined with machine learning. Precision refers to the quantity of correct data which is classified correctly by the classifier in classification process, while recall rate refers to the quantity of data that can be found in all the correct data. As the precision and recall rate of the classification process is a pair of contradictory measurements, the use of formula F1-Score can effectively balance the accuracy and recall rate, and the closer to 1 of F1-Score numerical value means better performance of classifier. Among them, TP, FP, TN and FN are used to respectively represent the real cases, false positive cases, real negative cases and false negative cases, and the calculation precision as well as recall rate along with F1-score formula are as follows:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

$$P = \frac{TP}{TP + TN} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

$$F1 - Score = \frac{2PR}{P + R} \tag{13}$$

At the same time, we use confusion matrix to illustrate the performance in the family classification by using memory dump grayscale as classifier features, and the overall detection performance of the classifier is revealed by using the ROC curve.

Experimental comparison

In the experiment, we measure the correct classification of data sets in 4.1 by using percentage. Our experiment is compared with SPAM-GIST (Nataraj and Manjunath, 2016) and HPC (Demme et al., 2013), meanwhile, we use memory dump grayscale through k-NN classifier (k = 3; K = 5), random forest (RF), neural network (MLP). Through 10-fold cross-validation, the experiment selects the optimal result to compare it with the previous research of the other two groups, and the experimental result shows the accuracy of the overall classification result, see Table 2:

From the above table, it can be seen that using MLP can have higher detection rate and good F1-Score performance, and from the comparison of experimental results, it can be seen that the SPAM-GIST method (Nataraj and Manjunath, 2016) can reach 95% or above of detection accuracy in theoretical environment experiment. When this method is directly applied to malware in the wild, it can only reach 71.9% of detection accuracy which is because after the malware uses encryption and code padding technologies, static detection cannot accurately detect malware. From the experimental result, it can be seen that the result of using memory dump grayscale for classifying malware is better than the HPC (Demme et al., 2013). During the recurrence of the research of Demme et al., we found that although Microsoft officially provided API interfaces for the performance counter obtained under Windows system, the sampling accuracy cannot be accurate to the instruction cycle like in article (Demme et al., 2013; Ozsoy et al., 2015, 2016), and the sampling interval is much larger than the above research. However, good accuracy is still reflected in the actual test. The performance counter we set in sandbox for reading corresponding process would perform sampling once every 500 ms. After the completion of sandbox operation, intercepting the first 512 data as feature vectors and using k-NN, RF as well as MLP to classify them. The optimal detection result can reach 85.9% while the average detection rate is around 80%. This is because, under Windows platform, the code which is different from that of experiment of article (Demme et al., 2013)is merged into Linux kernel so that it can accurately perform interception and sampling according to instruction cycle. And the ARM with open enough system on chip

Table 2
Comparing experimental result with SPAM-GIST and MAP.

Method	Accuracy	Precision	Recall rate	F1-Score
SPAM-GIST	71.9%	72.3%	70.1%	71.2%
HPC	86.2%	85.9%	74.5%	79.8%
k-NN(k = 3)	88.7%	89.1%	88.7%	88.9%
k-NN(k = 5)	89.9%	90.3%	89.6%	89.9%
RF	93.9%	93.7%	93.4%	93.5%
MLP	95.2%	95.1%	93.2%	94.1%

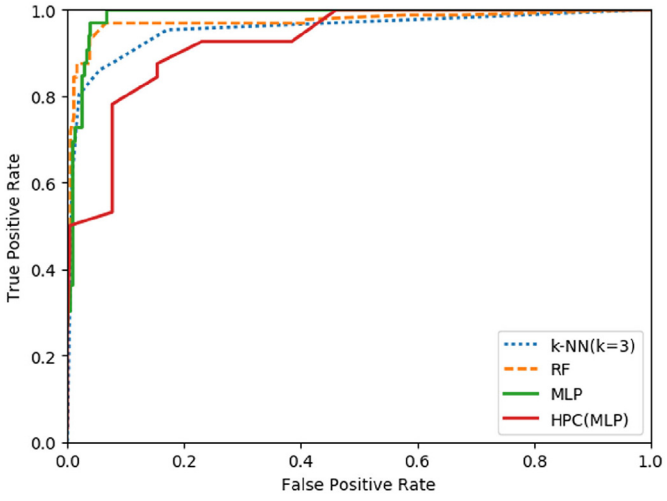

Fig. 5. ROC curves of each experiment.

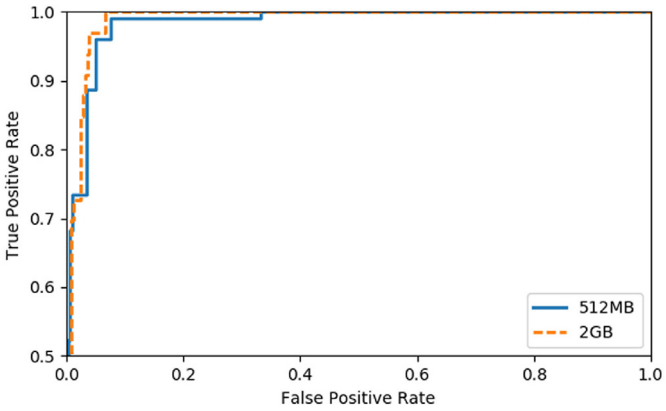

Fig. 6. ROC curves for guest machine of different RAM size.

(SoC) can also sample data according to the above conditions. Our sampling size on Windows system is too thick which may be the main reason why the accuracy is not as good as article (Demme et al., 2013).

Through ROC (receiver operating characteristic) curve and AUC (Area Under Curve)area, the horizontal coordinate of x-axis for ROC curve image is false positive rate (False Positive Rate), and the vertical coordinates of y-axis is true positive rate (True Positive Rate), as shown in Fig. 5. In the figure, under the condition that different false positive rates are took as threshold, the classification result by using MLP for detecting malware memory dump grayscale shows the best as 6% of false positive rate could be up to 99.9% of true positive rate, whereas using random forest and k-NN requires a higher false positive rate to increase the true positive rate.

When the guest machine memory is set to 512MB, only the MLP is used to classify the extracted memory dumps, and the classification result shows that the precision is not significantly decreased. The dump file classification results of different RAM size are shown in Fig. 6.

The above is the comparison of experimental detection result between the ROC curve of various average family classifications and HPC under Windows platform, in order to verify the accuracy of using malware memory dump grayscale on each family classification. This experiment uses different classifiers mentioned above to evaluate experimental result with combination of confusion matrix. In the ideal state, all the numerical value of matrix diagonal is 1 and the rest is 0. The closer to diagonal matrix for the numerical value of actual classifier detection result means better classification results of the classifier. As shown in Fig. 7, upon the closer to 1 of the classification result for each category we set in experiment, the color is darker. It can be seen from the figure that the classification result by using multilayer perceptron (MLP) in Fig. 7(d) is the best with max precision up to 95%.

Discussion

By using the memory dump as a feature, it is found in experiment that, when memory is large enough, the page swapping between memory and virtual memory cannot be very frequent with enough available data effectively stored in memory. Using grayscale images can convert similar binary data into similar textural features so as to effectively detect the malware.

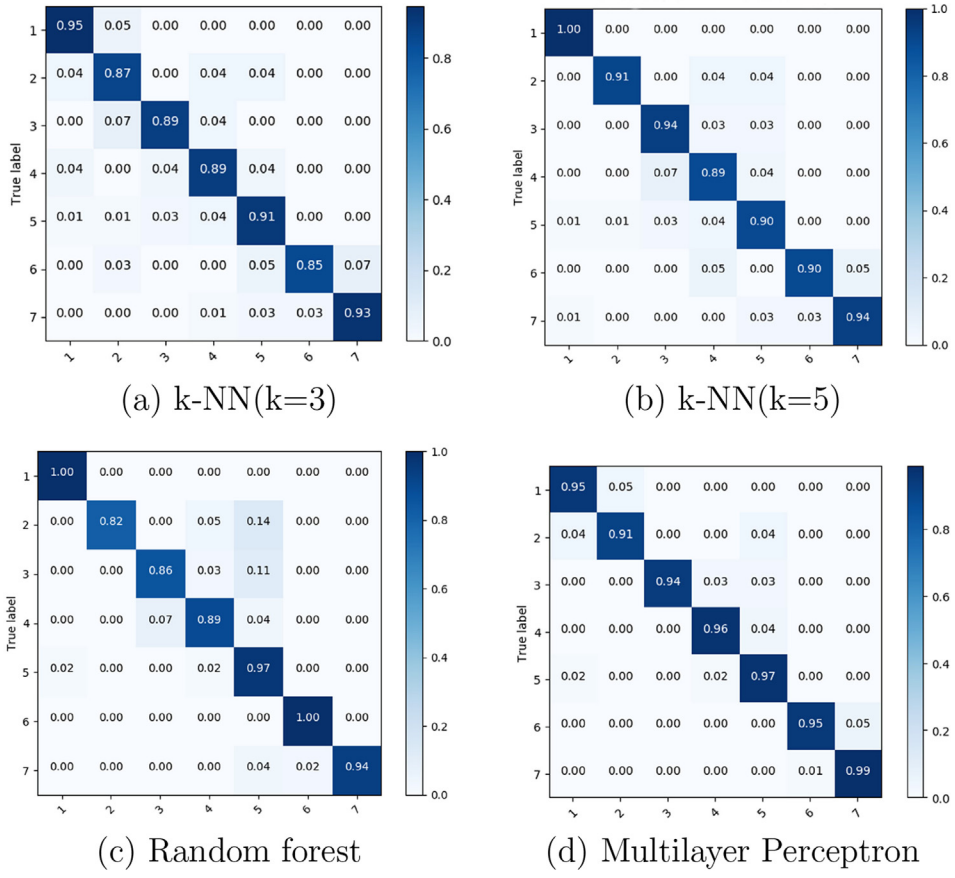

Fig. 7. Comparing the results of using various classifier to classify different family confusion matrix.

In addition, now the researches of using hardware features for detecting malware (Ozsoy et al., 2015, 2016) that use hardware features for detecting malware adopt hardware monitoring system operating environment, but there are certain disadvantages under hardware environment, for example, the identification of hardware driver, hardware PID, VID and other systems may escape from detected features as malware. Although the software has some vulnerability in nature, it can provide a variety of confusion methods against detection so as to avoid the environmental detection of malware.

Conclusion

The current dynamic analysis technology based on software features cannot effectively cope with malware evasion, and there is still some room for improvement for dynamic analysis method based on hardware features in aspect of classification accuracy. Aiming at this problem, this paper proposes a dynamic analysis method based on memory dump data. To dump the malware memory of runtime to binary files, then through binary files to convert it into grayscale images, to convert images into images of a specific size, and then to implement classification by using the multilayer perception method. Through the experimental verification of malware data sets, it shows that the method proposed in this paper is slightly higher on accuracy than the recently proposed method based on other hardware features. We found in the experimental process that there is a certain correlated characteristic in memory data between different processes. In the future work, we will continue to study the analysis method of malware hiding in full memory space of system as well as the defense research for fileless malware.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61571364), and the Shaanxi Provincial Natural Science Foundation (No. 2017JM6037).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.diin.2018.09.006.

References

Andronio, N., Zanero, S., Maggi, F., 2015. Heldroid: dissecting and detecting mobile ransomware. In: International Workshop on Recent Advances in Intrusion Detection, pp. 382–404.
Arp, D., Spreitzenbarth, M., Hbner, M., Gascon, H., Rieck, K., 2014. Drebin: effective and explainable detection of android malware in your pocket. In: Network and Distributed System Security Symposium.
Case, A., Iii, G.G.R., 2016. Detecting objective-c malware through memory forensics. Digit. Invest. 18, S3–S10.
Cuckoo sandbox, https://cuckoosandbox.org/.
Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision & Pattern Recognition, pp. 886–893.
Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., Stolfo, S., 2013. On the feasibility of online malware detection with performance counters. In: International Symposium on Computer Architecture, pp. 559–570.
Egele, M., Scholte, T., Kirda, E., Kruegel, C., 2008. A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. 44 (2), 1–42.
Y. Feng, O. Bastani, R. Martins, I. Dillig, S. Anand, Automated Synthesis of Semantic Malware Signatures Using Maximum Satisfiability.
N. Idika, A. P. Mathur, A Survey of Malware Detection Techniques, Purdue University.
Javaheri, D., Hosseinzadeh, M., 2017. A framework for recognition and confronting of obfuscated malwares based on memory dumping and filter drivers. Wireless Pers. Commun. 98 (4), 1–19.
Kaspersky lab, 2016. Detects 360,000 New Malicious Files Daily up 11.5% from (December 2017). https://www.kaspersky.com/about/press-releases/2017_kaspersky-lab-detects-360000-new-malicious-files-daily/.
N. Ketkar, Stochastic Gradient Descent, Optimization.
Khasawneh, K.N., Ozsoy, M., Donovick, C., Abu-Ghazaleh, N., Ponomarev, D., 2015. Ensemble Learning for Low-level Hardware-supported Malware Detection. Springer International Publishing.
Kirat, D., Vigna, G., Kruegel, C., 2011. Barebox:efficient malware analysis on bare-metal. In: Twenty-seventh Annual Computer Security Applications Conference, ACSAC 2011, Orlando, FL, USA, 5-9 December 2011, pp. 403–412.
D. Kirat, G. Vigna, C. Kruegel, Barecloud: bare-metal analysis-based evasive malware detection, Malware Detection.
Korkin, I., Nesterov, I., 2015. Applying memory forensics to rootkit detection. In: The ADFSL Conference on Digital Forensics. Security and Law.
Liu, J., Song, J., Miao, Q., Cao, Y., 2013. Fenoc: an ensemble one-class learning framework for malware detection. In: Ninth International Conference on Computational Intelligence and Security, pp. 523–527.
X. Lu, Z. Duan, Y. Qian, W. Zhou, Mcdf:a malware classification method based on deep forest, Journal of Software, CNIn press.
Nataraj, L., Manjunath, B.S., 2016. Spam: signal processing to analyze malware [applications corner]. IEEE Signal Process. Mag. 33 (2), 105–117.
Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J., 2011. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: ACM Workshop on Security and Artificial Intelligence, pp. 21–30.
Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011. Malware images:visualization and automatic classification. In: International Symposium on Visualization for Cyber Security, pp. 1–7.
New global cybersecurity, February 2018. Report Reveals Cybercrime Takes Almost $600 Billion Toll on Global Economy. https://www.mcafee.com/us/about/newsroom/press-releases/press-release.aspx?news_id=20180221005206/.
Openmalwarebenchmark, http://malwarebenchmark.org/, accessed April 5, 2018.
Ozsoy, M., Donovick, C., Gorelik, I., Abughazaleh, N., Ponomarev, D., 2015. Malware-aware processors: a framework for efficient online malware detection. In: IEEE International Symposium on High PERFORMANCE Computer Architecture, pp. 651–661.
Ozsoy, M., Khasawneh, K.N., Donovick, C., Gorelik, I., Abughazaleh, N., Ponomarev, D.V., 2016. Hardware-based malware detection using low level architectural features. IEEE Trans. Comput. 65 (11), 3332–3344.
Procdump, https://docs.microsoft.com/sysinternals/downloads/procdump/.
Z. Ramzan, V. Seshadri, C. Nachenberg, Reputation-based Security: an Analysis of Real World Effectiveness.
Smutz, C., Stavrou, A., 2016. When a tree falls: using diversity in ensemble classifiers to identify evasion in malware detectors. In: Network and Distributed System Security Symposium.
Tang, A., Sethumadhavan, S., Stolfo, S.J., 2014. Unsupervised Anomaly-based Malware Detection Using Hardware Features 8688, pp. 109–129.
Virustotal, https://www.virustotal.com/.