A Malware Variant Detection Method Based on Byte Randomness Test

Shuhui Qi, Ming Xu and Ning Zheng
Internet and Network Security Laboratory, Institute of Computer Science, Hangzhou Dianzi University
Hangzhou, China
Email: shuhuiqi @yeah.net, {mxu, nzheng}@hdu.edu.cn

***Abstract*—Malware variants, referring to the different members in the same malware family, are generally produced by simply modifying the existing malwares in order to avoid being detected by the traditional signature-based methods. The mass of malware variants has brought great difficulties to detect malicious code. In this paper, a malware variants detection method based on byte randomness tests is proposed. The bytes distribution value of the instruction sequences obtained from randomness tests , named as the byte randomness profiles, can preserves enough local detail about program, so it can be used as feature vector to represent malware in a distinctive manner. Moreover, the sum of squares distance (SSD) and the cosine similarity (COS) are used to measure the distinctiveness between two malwares. Experimental results show that the proposed method provides a fast and effective way to detect variants of known malware families.**

***Index Terms*—instruction sequences, byte randomness profile (BRP), feature vector, SSD, COS**

I. INTRODUCTION

Malware, short for malicious software or malicious code, is defined as: Programming code that is capable of causing harm to availability, integrity of code or data, or confidentiality in a computing system encompasses Trojan horses, viruses, worms, and trap-doors [1]. With the wide application of computer in the modern information society, it has become a serious threat to the security of networks. In recent years, the quantity and the diversity of malware are increasing sharply, which brings up a great challenge for antivirus work.

In order to make much more new malwares fast and simply, malware maker often modify the existing malwares based on their original files to produce new ones which called malware variants. Self-mutating [2] is generally used to produce malware variants by changing the contents of binary files of malwares while preserving their malicious behavior. Mutation Techniques include instructions substitution, instructions recording, garbage insertion, variable substitutions and so on. Another method to make variant is code reusing, malware maker manually copied previously existing code to new one, which uncovers code sharing among malware variants. However, such approaches to produce malwares brings great difficulties for the identification of malwares for the reason they not only do change the content of the malware files, especially the signature bytes, which enables malwares to bypass the antivirus software [2], but also make the number of malware increase dramatically.

In this paper a byte randomness profile based detecting model is developed to deal with the issue of malware variants detection. The issue means to mine the similarity on content, behavior and function between the different programs, and then determine whether they are variants with each other. In the proposed model, the similar between two malware instances is computed in terms of byte randomness profile which is indicated by the sum-of-squares distance (SSD) and the cosine similarity (COS). If having a high similar, they are regarded as variants of each other. The proposed method first disassembles the sample file, extracts the instruction sequences and saves it in hexadecimal; second, the hexadecimal file is used to calculate byte randomness profile which indicates the feature of malware; finally, we provide a series of experiments to evaluate the ability of this method for malware variant detection.

The rest of this paper is organized as follows. Section II generally discusses the related work. The details about proposed method are described in section III. Section IV presents and discusses the experimental result. In section V, the limitations and future work are pointed outs. Section VI concludes the paper.

II. INTRODUCTION

In order to hinder the crazy spread of malware effectively, many researchers have proposed a lot of methods to analysis and detect malware. They mainly fall into two categories, static analysis and dynamic analysis.

The method based on static analysis generally uses software to disassemble malware files without their execution, and adopts byte-level [3][4], function-call graph [5][6], instruction operation codes sequence[7][8] or string feature[9] to extract signature, which represents structure and function of malware. In the paper [4], malwares are analyzed using two approaches: disassembly, utilizing IDA-Pro, and the application of a

This paper is supported by NSFC (No. 61070212 and 61003195), Natural Science Foundation of Zhejiang Province, China (No. Y1090114), the State Key Program of Major Science and Technology (Priority Topics) of Zhejiang Province, China under (No 2010C11050).
Corresponding author: nzheng@hdu.edu.cn

doi:10.4304/jcp.8.10.2469-2477

dedicated state machine in order to obtain the set of functions comprising the executables based on byte-level. Shanhu Shang [5] uses a novel algorithm to construct the function-call graph from the assembly instructions, and proposes an effective graph matching method based on vertexes to compute similarity between two binaries. The paper [6] is an improvement of [5]. Specially, Yanfang Ye,et al [10] have developed an intelligent malware detection system (IMDS) by rule learning based on the static analysis of application programming interface (API) execution calls. In their later work [11], they use some methods to reduce the number of generated rules, which make it easy and effective for their method. But there exist lots of limits of static analysis [12], the authors describes the limitation of static extraction methodology and demonstrates that static extraction alone is not enough detect malicious code.

The limits of static analysis [12] prompted researchers and practitioners to develop automated, dynamic malware analysis systems. In contrast to static analysis, dynamic techniques execute malware in a simulated environment and analyze its interaction with the system, including API sequence [13], system call sequence [14][15][16] and state changes of the system[17]. In the paper [13], they extract API calls as behavioral features from executables in a virtual environment and apply pattern recognition algorithms and statistical methods to differentiate malware from cleanware. Gérard Wagener [14] proposes a flexible and automated approach to extract malware behavior by observing all the system function calls performed in a virtualized execution environment. M. Baileyb [17] proposed a new classification technique that describes malware behavior in terms of system state changes rather than in sequences or patterns of system calls, using normalized compression distance (NCD) to automatically categorizing these profiles of malware into groups.

To summarize, static analysis is fast and safe without executing malware sample. It can cover the entire malware code, not missing information, but it has difficulty detecting unknown malware because of the use of code obfuscation. However, dynamic analysis has some defects as well. It needs too much time and resources. When collecting the traces of malware in a virtual environment, the threshold of time is difficult to determine. Meanwhile, Dynamic analysis has been facing the challenge of anti-tracing, anti-debugging and anti-emulating, so it cannot capture the real behavioral characteristics of malwares.

In this paper, the model is based on static analysis and our concern is about identification and detection of the malware variants. The malware variants are mainly produced by self-mutating [2] or code reusing, others are not considered. Although the malware variants change the signature, which enables the traditional signature-based detection methods to fail, but most of them change only tens of bytes and share most of common code. This little change results in that the byte randomness profile varies lightly on the statistical level. For example, the code reordering alters the sequence of two and more instructions, but its bytes distribution [18] keeps the same. Other self-mutating technologies and code reusing do change the content of malware in byte level, however, the alteration only makes up a small proportion of the entire file, which has almost no influence on byte randomness profile distribution. This can be demonstrated in scanning experiments in Section IV. Therefore, the byte randomness profile of malware files is chosen as feature vector to identify and detect malware variants.

The main idea of the proposed method was inspired by [18] and [19]. Sheng Yu [18] proposed a novel byte frequency refers to the frequency of the different unsigned bytes in the corresponding binary file to deal with the malware variants identification. We continue to collect the byte randomness profile based upon byte frequency [18] using two randomness test algorithms [19], and then utilize two metrics, the *SSD* and *COS*, to calculate the similarity between two malware samples.

The randomness test, sometimes called ‘entropy’ tests, can be a fast way to describe the information distribution of the tested object. Ebringer et al. developed a randomness test [19] that preserved local detail. Li Sun [20] employed a refined version of the sliding window randomness test with pattern recognition techniques for the classification of malware packers. However, our method is different from [20]. On one hand, their inputted file was the whole file in the form of “exe” while instruction sequence file produced by disassembly software is used in this paper. On the other hand, they were considering identification of the packers, rather than malwares.

III. Description of Methods

A. The Detection Model Overview

The goal of the proposed model is to compare the similarity between two malware samples based on their byte randomness profile. It is a multi-step process. It consists of samples preprocessing, constructing byte frequency, extracting of the Byte Randomness Profile (BRP) and testing similarity.

- **Preprocessing Sample**. It uses the disassemble tool software to disassemble the malware sample, extracts its pure instruction sequences, and produce byte distribution files about its operation instructions in hexadecimal. All files are composed by tons of bytes.
- **Constructing Byte Frequency Array**. The byte distribution files are inputted to calculate its byte frequency array. This process is similar as [18] proposed. A sample’s byte frequency array is denoted as $M = \{m_1, m_2, ..., m_{254}\}$, where m_i indicates how many bytes in the sample are of value i (i=1, 2…, 254). Except for the bytes of value 0 and 255 which are usually used as filling bytes, there are 254 kinds of bytes of different values.
- **Extracting of the BRP**. In this step, the BRP of sample is extracted based on the output provided by the previous step using two randomness

scanning algorithm presented by [19]. The byte randomness profile is finally denoted as *n*-dimensional vector space.

- **Testing Similarity**. Finally, the two measure metrics, COS and SSD, are used to evaluate the similarity between two malware instances based on byte randomness profile. The whole procedure is described as in the algorithm 1.

Algorithm 1 Similarity Comparison Algorithm

Input: the testing instance's assembly file
Output: the similarity between two samples
01 Read the file in the form of hexadecimal
// constructing byte frequency array
02 **while** not the end of file
03 **for** each byte
04 *FreArray*[*byte*] + = 1;
05 **end for**
06 **end while**
/* extracting of the byte randomness profile through two randomness tests */
07 **for** each sample
08 *BRP*[*i*] = randomness_test(*assembly file, FreArray*);
09 **endfor**
/* evaluating the similarity between two samples in a pairwise fashion */
10 **for** each sample
11 **for** each sample
12 *cos_sim* = cos(*BRP*[*i*], *BRP*[*j*]);
13 *ssd_dis* = ssd(*BRP*[*i*], *BRP*[*j*]);
14 **endfor**
15 **endfor**
16 return *con_sim, ssd_dis*

B. Byte Randomness Profile

The byte randomness profile is introduced as the characteristic of the malware which represents the malware in a distinctive way. The byte randomness profile simply gives an overall byte randomness value for the entire file. There are two randomness scanning test algorithms. The first is a 'sliding-window' approach which is dependent on the size of file, and thus less convenient for comparative purpose. The second generates a fixed number of feature vectors, and is fit to lager files. Both algorithms require that the Huffman tree be constructed for the entire file. More details about the both approaches were proposed by [19].

C. The Metrics of Proximity between Malwares

Here, the byte randomness profile of malware sample is an *n*-dimeional vector space, this paper presents two commonly used measures. They are sum-of-squares distance (*SSD*) and the cosine similarity (*COS*), the *SSD* measures the distance between two vectors, and the *COS* measures the angle. Given two vectors $\overrightarrow{M}$ and $\overrightarrow{N}$, $\overrightarrow{M}=\{m_1,m_2,...m_i,...\}$ and $\overrightarrow{N}=\{n_1,n_2,...n_i,...\}$, where the m_i, n_i indicate the *i*th eigenvalue of two vectors. The *COS* and *SSD* are represented as follows respectively:

$$COS\left(\overrightarrow{M},\overrightarrow{N}\right)=\sum m_i n_i \Big/ \sqrt{\sum m_i^2 \sum n_i^2} \quad (1)$$

$$SSD\left(\overrightarrow{M},\overrightarrow{N}\right)=\sqrt{\sum\left(m_i-n_i\right)^2} \quad (2)$$

The SSD and COS should be obtained in a pairwise fashion to evaluate the similarity between two samples, they are regarded as in one malware family when $SSD\left(\overrightarrow{M},\overrightarrow{N}\right)\le\alpha$ and $COS\left(\overrightarrow{M},\overrightarrow{N}\right)\ge\beta$, where α and β are the thresholds. In order to obtain them, a training experiment was implemented. From the results, it is noted that the four values listed in TABLE.I perform better, thus, we adopt these standards to determine whether the testing samples are variants.

D. Evaluation Metrics

TABLE I.
THE LIST OF THRESHOLDS

Algorithm	threshold	value
Sliding window	*COS*	0.998
	SSD	600
Fixed sample count	*COS*	0.9985
	SSD	10000

Algorithm 2 PRAF Computing Algorithm

Input: two testing malware families *X* and *Y*
// X has *m* samples and *Y* has *n* samples
Output: the PRAF value between two malware families
01 *mn* = 0, *PRAF* = 0.00, *SimValue* = 0.00
02 **for** each sample x_i in *X*
03 **for** each sample y_j in *Y*
04 **if** (x_i.*label* == y_j.*label*)
/* judging whether they are from the same family, it is the correct reference. */
05 *IsFamily* = True
06 **else** *IsFamily* = False
07 **endif**
08 *SimValue* = SCA (x_i,y_j)
/* evaluating the *cos_sim* between two samples using Algorithm 1 */
09 **if** (*SimValue* >= *COS*)
/* updating the value of *IsFamily* labeled as *IsFamily_update*, it is the testing result */
10 *IsFamily_update* = True
11 **else** *IsFamily_ update* = False
12 **endif**
13 if (*IsFamily* == *IsFamily_update*)
// counting the number of correct results
14 *mn*++
15 **endif**
16 *PRAF*=2**mn*/((*m*+*n*)*(*m*+*n*-1))
// calculating the value of *PRAF* using (3)
17 **endfor**
18 **endfor**
19 return *PRAF*

In the experiment, it is estimated the ability of detect malware variant. In order to implement it, this paper proposed a new evaluation metrics named as PRAF.

Given two malware families *X* and *Y*, *X* has *m* malwares and *Y* has *n* malwares, they are denoted as $X=\{x_1, x_2 \ldots, x_m\}$ and $Y=\{y_1, y_2 \ldots, y_n\}$, and then *SSD* and *COS* are calculated by pairwise to determine similarity among the (*m*+*n*) malware samples which are from two different malware families. The ideal result is that x_i and x_j are in the same group, y_i and y_j are in the same group, and x_i and y_i are in the different group. We count the number of this result as *mn*, so the Precision Rate among Families (PRAF) is defined as:

$$PRAF = 2\times mn \Big/ \Big((m+n)\times(m+n-1)\Big) \qquad (3)$$

Here, *m* and *n* denotes the number of members in two families *X* and *Y* respectively and *mn* counts the value of correct results, it is a new variable. The pseudo-code for PRAF calculation is given in algorithm 2, it employs *COS* as the metric of similarity. The procedure based on *SSD* is similar to algorithm 2. In the fact, it can be implemented by using *SSD* and *DisValue* to replace *COS* and *SimValue* respectively, and when *DisValue* is not higher than *SSD*, they are labeled as in one family.

IV. Experimental and Results

The experiments are presented to evaluate the performance of the proposed methods. The more details are following.

A. Malware Dataset

First the malware samples used in the experiments were downloaded from VX Heavens [21], as shown in TABLE.II. Because of the randomness tests are largely dependent on the size of input files, so it lists the average file size (AFS) of each malware family in the last column. But the files are the byte distribution files that outputted in the first step rather than the original files.

B. Analysis of Experiments Results

1) Byte Randomness Profile Scanning

Experiments have been carried out intensively on three malware families, we randomly select three samples in each family: *Virus.Win32.Evol*, *Virus.Win32.Mental* and *Email-Worm.Win32.Sober*. The sizes of these files range from 18K to 139K. The files that are over 100KB are labeled as big files and the small set comprises files smaller than 100KB. In the fixed sample count algorithm, the sample count s is set to 1024 to balance small and big file. In the sliding window algorithm, the window size *w* is set to 32 bytes and the skip size *a* is 32. These is two reasons for this based on its procedure described in [19]. The attribute value and size of features are directly related to the *w* and *a*. On the one hand, we set *w* be equal to *a*, so that there is no overlap value between windows, avoiding the repeating calculation. On the other hand, if we set them as 16, it is seen that some malware instances' feature size is higher 10000, it's too big and takes more time. When they are decided as 64, some malware instances' feature size reduces to lower 100 unexpectedly, hiding the more details information. In order to balance the relationship between attribute value and size, this paper selects 32. So, the size can be controlled in a reasonable range, at the same time, it can keep more details as much as possible. The final step is to plot the output. Fig.1-3 show plots of three malware samples indicated as (a), (b) and (c) in different malware families using the sliding window algorithm, x-axis indicates the number of the byte randomness profile and y-axis represents the normalized value, which reflects the byte distribution of instruction sequences about malware. Similarly, Fig.4–6 show plots of three malware samples in different malware families using the fixed sample count algorithm.

TABLE II.
SUMMARY OF MALWARE FAMILIES FOR EXPERIMENT

ID	Name	Malware Type	No.	AFS
1	Evol	Virus.Win32	3	18K
2	Klez	Email-Worm.Win32	10	125K
3	Mental	Virus.Win32	4	17K
4	Mimail	Email-Worm.Win32	9	36K
5	Netsky	Email-Worm.Win32	18	45K
6	Newbiero	Worm.Win32	6	371K
7	Roron	Email-Worm.Win32	17	163k
8	Sober	Email-Worm.Win32	14	139k
9	QQPass	Worm.Win32	32	69k
Total	9	N/A	113	N/A

The plots are made of an entire instruction sequences files showing areas of high randomness and low randomness. Note that the similar plots for belonging to the same malware family based on the byte randomness profile, even when compared by eye. It is clear that by this technique, different samples from different malware families seem to produce a distinctive plot. This suggests that the byte randomness profile can be used to represent feature of malware for further detection.

2) Malware Variants Detection

The byte randomness profile can be applied to evaluate the similarity between two malware samples. In this section, it used our evaluation standard-PRAF proposed in the Section III-D and COS and SSD presented in the (1) and (2). The result of experiment was shown in TABLE.III. It list the all the value of PRAF between any two malware families using two similarity measure COS and SSD in two byte randomness tests. The first column 'ComID' is the combination of two different malware family ID. For example, the number '12' delegates the combination of malware family one and malware family two that are noted in TABLE.II. Because of the sliding window algorithm, the byte randomness profile used as feature vector have different length between input samples, Ebringer et al.[6] developed four types of pruning. They were called First, Smallest, Ordered smallest and Trunk. We choose the 'First'.

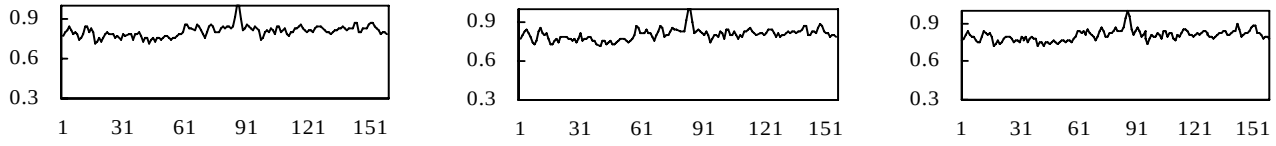

Figure 1. Sliding window randomness scan for Virus.Win32.Evol

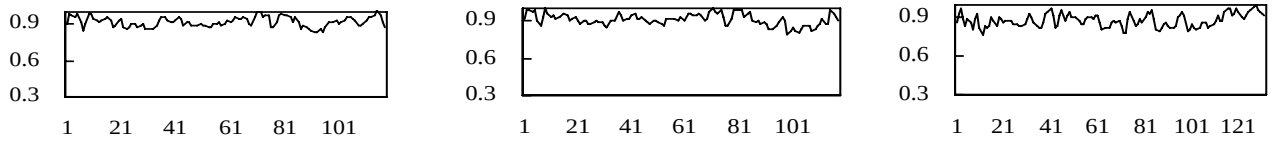

Figure 2. Sliding window randomness scan for Virus.Win32.Mental

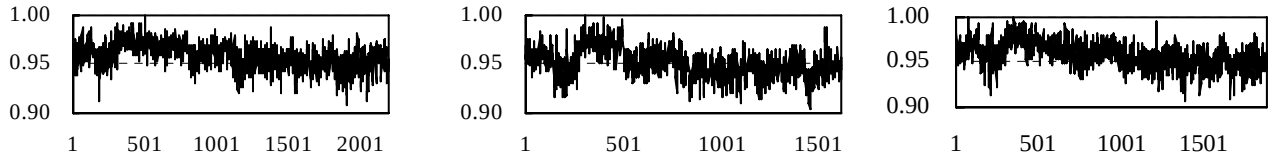

Figure 3. Sliding window randomness scan for Email-Worm.Win32.Sober

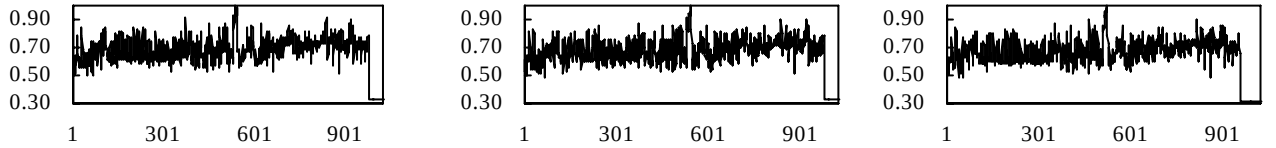

Figure 4. Fixed sample count randomness scan for Virus.Win32.Evol

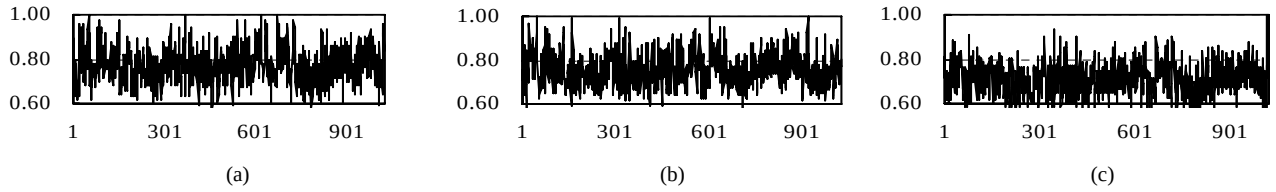

Figure 5. Fixed sample count randomness scan for Virus.Win32.Menta

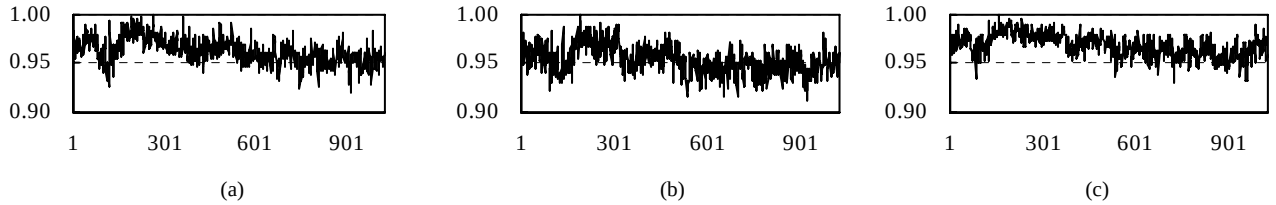

Figure 6. Fixed sample count randomness scan for Email-Worm.Win32.Sober

TABLE.III illustrates that different malware families have low similarity in the statistics level, especially the *Newbiero*, the PRAF between *Newbiero* and other malware families are all above 87%. In order to illustrate the conclusion that malware families exhibit some obvious differences, confusion matrix among malware families is shown in Fig.7, where dark color indicates high degree of confusion and light color low confusion. As seen in Fig.7, there exist some confusion between the families such as *Roron*, *Sober* and *QQpass*, other families have a little confusion. It does show that malware variant detection can be achieved by computing the byte randomness profile of the file. However, there were some failures of detection that were noted as 'N/A' labeled as TABLE.III. In the experiment, we found: the value of COS between two malwares in the same families was smaller than that between two malwares which are from different families, and the value of SSD between two

malwares in the same family was bigger than that between two malwares which are from different families. If any one of the two above phenomenon appears, it is a failure. The reason will be explained in the following.

TABLE III.
THE PRAF BETWEEN TWO MALWARE FAMILIES

ComID	Sliding window(First)		Fixed sample count		The average of PRAF
	COS	SSD	COS	SSD	
12	83.33%	*N/A*	96.15%	100.00%	93.16%
13	85.71%	85.71%	61.90%	85.71%	79.76%
14	87.88%	87.88%	92.42%	74.24%	85.61%
15	90.95%	89.05%	76.19%	63.33%	79.88%
16	94.44%	*N/A*	66.67%	100.00%	87.04%
17	96.32%	*N/A*	98.42%	100.00%	98.25%
18	98.53%	100.00%	97.79%	83.82%	95.04%
19	81.51%	*N/A*	90.25%	80.67%	84.15%
23	73.63%	*N/A*	93.41%	100.00%	89.01%
24	47.37%	66.67%	50.88%	81.87%	61.70%
25	52.38%	74.34%	48.15%	79.63%	63.62%
26	87.50%	87.50%	87.50%	100.00%	90.63%
27	52.14%	54.13%	51.57%	81.20%	59.76%
28	84.06%	85.14%	49.28%	57.97%	65.45%
29	59.81%	66.09%	56.33%	64.92%	61.79%
34	85.90%	84.62%	88.46%	89.74%	87.18%
35	88.31%	86.15%	76.62%	69.70%	80.19%
36	100.00%	*N/A*	73.33%	100.00%	91.11%
37	82.38%	*N/A*	97.14%	94.76%	91.43%
38	98.04%	72.55%	96.08%	84.31%	87.75%
39	69.37%	*N/A*	93.65%	79.68%	80.90%
45	55.56%	54.13%	49.86%	52.14%	52.92%
46	85.71%	85.71%	78.10%	100.00%	87.38%
47	56.62%	*N/A*	63.38%	96.31%	72.10%
48	52.57%	*N/A*	68.38%	80.63%	67.19%
49	37.56%	32.80%	42.44%	49.02%	40.46%
56	91.67%	82.97%	77.17%	100.00%	87.95%
57	50.92%	45.21%	49.41%	89.08%	58.66%
58	50.60%	*N/A*	52.42%	83.27%	62.10%
59	52.24%	*N/A*	47.84%	56.73%	52.27%
67	94.07%	92.09%	94.07%	83.79%	91.01%
68	92.11%	92.11%	92.11%	84.21%	92.79%
69	94.03%	97.87%	89.90%	100.00%	95.45%
78	48.60%	52.90%	59.14%	55.91%	54.14%
79	55.19%	57.48%	48.98%	74.49%	59.03%
89	42.80%	58.74%	*N/A*	73.43%	58.33%

3) The Influence of file size differing ratio

It has been mentioned that the two versions of the randomness scanning algorithm are dependent on the size of input file, especially the sliding window approach, because the length of feature vector with different file size extracting from randomness test is varying. So we make a visualization in Fig.8 to illustrate the effect of file size differing ratio on the detection. First we use TABLE. IV to list the average file size differing ratio among different malware families. The 'ComID' means ID combination of one malware family and the other malware family that are noted in TABLE.I and DROF represents differing ratio of the average file size between two different malware families. In Fig.8, we can see the bigger the differing ratio of average file size between two different malware families, the better the detection rate. It also suggests all of the two versions of the randomness scanning algorithm are easily affected by the size of input files.

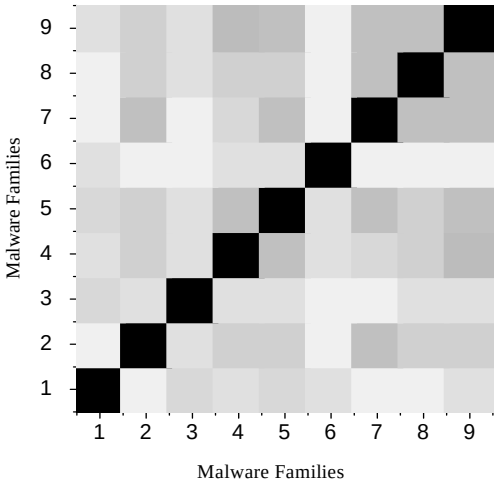

Figure 7. Confusion matrix among malware families, x-axis and y-axis indicate different malware families.

TABLE IV.
THE DIFFERING RATIO OF AVERAGE FILE SIZE

No.	ComID	DROF	No.	ComID	DROF
1	13	1.06	19	58	3.1
2	28	1.11	20	24	3.5
3	78	1.2	21	57	3.6
4	45	1.25	22	19	3.83
5	27	1.3	23	48	3.86
6	59	1.5	24	39	4.1
7	29	1.8	25	47	4.5
8	49	1.9	26	69	5.3
9	14	2.0	27	12	6.9
10	89	2.01	28	23	7.4
11	34	2.1	29	18	7.7
12	67	2.3	30	38	8.17
13	79	2.4	31	56	8.24
14	15	2.5	32	17	9.1
15	35	2.6	33	37	9.6
16	68	2.7	34	46	10.3
17	25	2.8	35	16	20.6
18	26	3.0	36	36	21.8

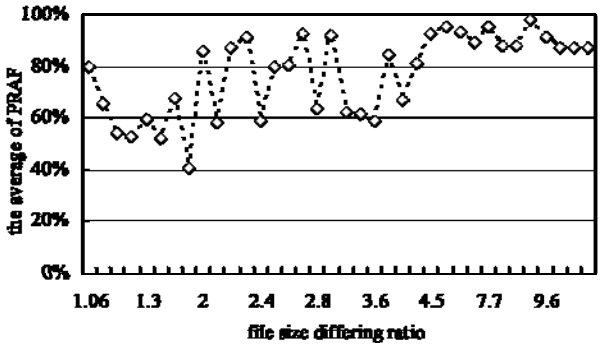

Figure 8. A Visualization about the Influence of file size. X axis--file size differing ratio, Y axis—the average of PRAF.

4) The Evaluation of two Similarity Measures

In order to explore the ability of the two similarity measures, we plot the PRAF based on SSD and COS used in two randomness algorithms respectively. Fig.9

illustrates the results of experiments. From the plot, the fixed sample count algorithm is better than the sliding window test with the 'First' pruning algorithm. Through the sliding window test, the byte randomness profile that used as last feature vector to compare have different length, for example, the average length of feature vector about malware family *Virus.Win32.Evol* is 158 while that is all above 1600 in malware family *Email-Worm.Win32.Sober*, but the last length to compare is 158, so it loses lots of information and brings about the failure of detection, we denote it as 'N/A'. However, in the fixed sample count test, the length is fixed and unified, not missing information, so the fixed sample count test has only a detecting failure while several failures appear in the sliding window test. It is also noted that the SSD measure outperforms the COS measure in most of experiments.

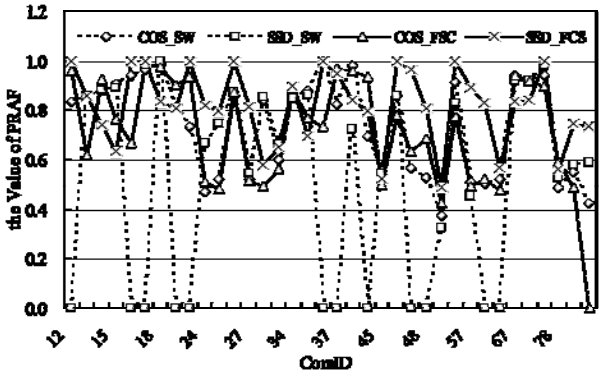

Figure 9. A Visualization about Evaluation of two Similarity Measures. X axis—ComID, Y axis—the value of PRAF. a - COS in SW (Sliding Window), b-SSD in SW, c-COS in FSC (Fixed Sample Count), d-SSD in FSC.

TABLE V.
THE BENIGN AND MALICIOUS SAMPLES

Program Name	Description
ipv6.exe	system file from system32 folder, Win 7 SP1
lsass.exe	system file from system32 folder, Win 7 SP1
netstat.exe	system file from system32 folder, Win 7 SP1
bootcfg.exe	system file from system32 folder, Win 7 SP1
cacls.exe	system file from system32 folder, Win 7 SP1
comp.exe	system file from system32 folder, Win 7 SP1
md5sum.exe	md5 checksum [22]
orm.Win32.Mimail.e	a potent worm spreads via email attachment
Worm.Win32.Mimail.f	a variant of the Mimail family
Worm.Win32.Mimail.g	a variant of the Mimail family
Virus.Win32.Evol.a	a 32-bit metamorphic virus
Virus.Win32.Evol.b	a variant of the Evol family
Virus.Win32.Evol.c	a variant of the Evol family
Worm.Win32.Roron.25b	a potent worm spreads via email attachment
Worm.Win32.Roron.31	a variant of the Roron family
Worm.Win32.Roron.33	a variant of the Roron family

5) *Evaluation on Hybrid Samples*

In this section, we apply byte randomness profile to measure the similarities between malware and benign binaries. The data set used in this experiment consists of 7 legitimate binaries which are collected from the system files and application software and 9 malicious programs from three malware families respectively. The details about them are listed in TABLE.V.

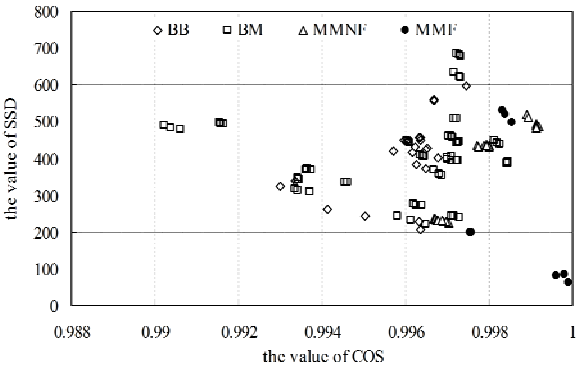

Figure 10. The Result among hybrid binaries using Sliding Window Algorithm. X axis—COS, Y axis—SSD.BB- Benign VS Benign, BM-Benign VS Malware, MMNF-Malware VS Malware not in the same family, MMF- Malware VS Malware in the same family.

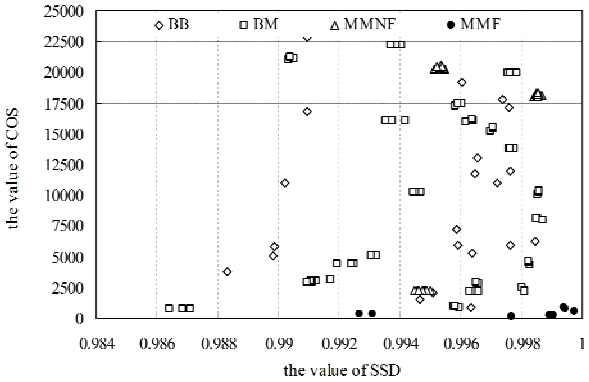

Figure 11. The Result among hybrid binaries using Fixed Sample Count Algorithm. X axis—COS, Y axis—SSD.BB- Benign VS Benign, BM-Benign VS Malware, MMNF-Malware VS Malware not in the same family, MMF- Malware VS Malware in the same family.

The SSD and COS between any two of these 16 binaries are respectively calculated by the two randomness test, producing 120 value pairs. Experimental results are pictured in Fig.10-11. From the Fig.10, benign programs don't have unified format or other correlations, so their distribution is irregular. The position among three malware families is distant, but samples' positions in the same family are intensive, it seems to a good clustering, but we haven't confirmed it. It is also noted that distinguishment between benign softwares and malwares is very obvious. The different numbers of the same family appearing in the lower right corner of area illustrates that the SSD between two malware from the same family is very small and COS is high, except the *Email-Worm.Win32.Roron*, the average file size of it is larger than others .We have mentioned that the sliding window algorithm is not suit to the lager files, and this phenomenon demonstrates it rightly. In the Fig.11, we can see the distribution is more irregular. In the fixed sample count algorithm, the feature of low randomness is masked or disappears in the big files with small sample count or over-represented in the small files with big

sample count. Because of this specialty, there are some pairs between malware and benign software, the SSD is smaller but COS is lower, or COS is higher but SSD is bigger. If using one similarity standard SSD or COS alone, the detection is not well, but we can obtain the ideal result when combinating SSD and COS simultaneously.

V. Limitation and Future Work

It is clear that byte randomness profile can be used to solve the issue of identification and detection of the malware variants from the preliminary experimental results. However, it has a few limitations. First, it does not handle encrypted/packed or polymorphic malwares. Second, other pruning methods should be employed to preserve more information when input samples have different size to improve the performance of siding window algorithm. What's more, it is necessary for some weights to be added to reduce the impact of input samples when files size is different. We should focus on these drawbacks in our future work, and vow to solve these issues.

In addition, this paper implements various experiments on known malware families only to testify the recognition capability of malware variants. As shown above, the byte scanning of an instance can provide a certain kind of 'characteristic' signal. Using this strategy, a malware family' signature is also represented as an n-dimensional vector. The value can be generated using the average value of a set of training data which are from the same family. Therefore, in the future, we should computer a byte randomness profile for each family and measure the similarity between the sample file and a family's signal to detect new variants.

VI. Conclusions

In this paper, our method collects the byte randomness profile of malware samples, the scanning plots suggest that the byte randomness profile can be used as the signature of the malware which represents the malware in a distinctive way in different families.

To compute the similarity between two binaries, this approach first converts the byte randomness profile into n-dimensional vector and uses two similarity metrics, SSD and COS to decide whether they are from the same family or not. In the end, we do our experiments in different samples sets: malware dataset from different malware families, and hybrid dataset consisting malwares and benign binaries. It not only shows that this method based on byte randomness test can be used for malware variants detection, but also can be employed to differentiate malware from cleanware and malware families classification. We also evaluate the robustness of byte randomness profile based upon different file sizes and the directivity of two similarity measures. Experiments show that two versions of the randomness scanning algorithm are easily affected by the size of input file.

References

[1] McGraw-Hill and Sybil P. Parker. McGraw-Hill Dictionary of Scientific and Technical Terms. McGraw-Hill Companies, Inc., 2003.

[2] Danilo Bruschi, Lorenzo Martignoni and Mattia Monga. Using code normalization for fighting self-mutating malware. Proceedings of International Symposium on Secure Software Engineering Washington, DC, USA, 2006.

[3] S.Momina Tabish, M. Zubair Shafiq and Muddassar Farooq. Malware Detection using Statistical Analysis of Byte-Level File Content. Proceedings of ACMCSI-KDD'09, June 28, 2009, Paris, France.

[4] Asaf Shabtai, Eitan Menahem, and Yuval Elovici. F-Sign:Automatic, Function-Based Signature Generation for Malware. IEEE Transactions on systems, man, and cybernetics—part c: applications and reviews, 2010.

[5] Shanhu Shang, Ning Zhen, Jian Xu, Ming Xu and Haiping Zhang. Detecting malware variants via function-call graph similarity. Proceedings of the 5th malicious and unwanted software (malware), 2010:113-120.

[6] Lingfei Wu, Ming Xu Jian Xu Ning Zheng and Haiping Zhang. A Novel Malware Variants Detection Method Based On Function-call Graph. Proceedings of the JICSIT 2011.

[7] Igor Santos, Felix Brezo, Javier Nieves,YosebaK.Penya, Borja Sanz, Carlos Laorden, and Pablo G. Bringas. Idea: Opcode-Sequence-Based Malware Detection. ESSoS 2010, LNCS 5965, pp. 35–43, Springer-Verlag Berlin Heidelberg 2010.

[8] Yanfang Ye ,Tao Li,Yong Chen and Qingshan Jiang. Automatic Malware Categorization Using Cluster Ensemble. Proceedings ACM KDD'10, July 25–28, 2010, Washington, DC, USA.

[9] Rafiqul Islam, Ronghua Tian, Lynn Batten and Steve Versteeg. Classification of Malware Based on String and Function Feature Selection. Proceeding of the 2th Cybercrime and Trustworthy Computing Workshop(CTC), IEEE 2010.

[10] Yanfang Ye,Dingding Wang,Tao L,Dongyi Ye and Qingshan Jiang. An intelligent PE-malware detection system based on association mining. J Comput Virol (2008) 4:323–334.

[11] Yanfang Ye, Tao Li, Qingshan Jiang, and Youyu Wang. CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,40(3):298 –307, may 2010.

[12] S.Momina Tabish, M. Zubair Shafiq and Muddassar Farooq. Limits of Static Analysis for Malware Detection. In ACSAC, pages 421–430. IEEE Computer Society, 2007.

[13] Ronghua Tian, Rafiqul Islam ,Lynn Batten and Steve Versteeg. Differentiating Malware from Cleanware Using Behavioural Analysis. Proceeding of the 5th International Conference on Malicious and Unwanted Software , 2010.

[14] Gérard Wagener, Radu State and Alexandre Dulaunoy, A. Malware Behaviour Analysis. Journal in Computer Virology, 4(4):279–287, November 2008.

[15] Hengli Zhao, Ning Zheng, Jian Li, Jingjing Yao,and Qiang Hou . Unknown Malware Detection Based on the Full Virtualization and SVM. Proceeding of the 2009 International Conference on Management of e-Commerce and e-Government, IEEE 2009.

[16] Mihai Christodorescu, Somesh Jha, and Christopher Kruegel. Mining specifications of malicious behavior. Proceedings of the 6th Joint Meeting of the European Software Engineering Conferenceand the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'07). New York, NY, USA: ACM Press, 2007, pp. 5–14.

[17] Michael Bailey, JonOberheide, Jon Andersen, Z.MorleyMao, Farnam Jahanian and JoseNazario. Automated classificationand analysis of internet malware. Proceedings of the 10th International Symposium on Recent Advances inIntrusion Detection (RAID'07), September 2007.

[18] Sheng Yu, Shijie Zhou, Leyuan Liu, Rui Yang and Jiaqing Luo. Detecting Malware Variants by Byte Frequency. Journal of Networks, VOL. 6, NO. 4, APRIL 2011.

[19] Tim Ebringer, Li Sun., Serdar Boztas. A Fast Randomness Test that Preserves Local Detail. Proceedings of 18th Virus Bulletin International Conference, 2008.

[20] Li Sun, Steven Versteeg, Serdar Boztas ,and Trevor Yann. Pattern Recognition Techniques for the Classification of Malware Packers. Proceedings of the ACISP 2010, LNCS 6168, pp. 370–390, 2010.

[21] VX heavens. http://vx.netlux.org, 2011.

[22] md5 checksum. http://www.etree.org/md5com.html, 2011.

Shuhui Qi received the Bachelor degree in Computer Network Engineering from Taiyuan Institute of Technology, Taiyuan, China, in 2010. Currently, he is studying in Hangzhou Dianzi University seeking for a Master degree of Computer application technology. His research interest includes Network security and malware analysis.

Ming Xu received the Ph.D. degree from Zhejiang University, China. He is currently a professor with Hangzhou Dianzi University, Hangzhou, China. His research interests include network security, digital forensics, content security, and information counterwork.

Ning Zheng received the Master degree from Zhejiang University, China. He is currently a professor and doctoral supervisor with Hangzhou Dianzi University, Hangzhou, China. His research interests include information management system and network information security.