Int. J. Inf. Secur. (2015) 14:1–14
DOI 10.1007/s10207-014-0242-0

REGULAR CONTRIBUTION

Malware analysis using visualized images and entropy graphs

Kyoung Soo Han · Jae Hyun Lim · Boojoong Kang · Eul Gyu Im

Published online: 29 April 2014

Abstract Today, along with the development of the Internet, the number of malicious software, or malware, distributed especially for monetary profits, is exponentially increasing, and malware authors are developing malware variants using various automated tools and methods. Automated tools and methods may reuse some modules to develop malware variants, so these reused modules can be used to classify malware or to identify malware families. Therefore, similarities may exist among malware variants can be analyzed and used for malware variant detections and the family classification. This paper proposes a new malware family classification method by converting binary files into images and entropy graphs. The experimental results show that the proposed method can effectively distinguish malware families.

Keywords Computer security · Malware analysis · Malware visualization

1 Introduction

The number of new malware and variants on the Internet has been continuously increasing with the help of various automated tools. Because of automated tools, similar modules may appear in malware variants, and these similar modules can be used to detect variants. Most of antivirus programs use certain character strings or patterns, i.e., signatures to detect malware [1]. However, signature-based detection methods can be avoided if diverse detection avoidance techniques are used in malware [2]. So, malware researchers have studied various analysis techniques to respond to malware variants and detection avoidance techniques [3–6].

This paper proposes a malware analysis method that uses visualized images and entropy graphs to detect and classify new malware and malware variants. Experimental results showed that our proposed method can effectively classify malware families.

The composition of this paper is as follows: Sect. 2 described related work on malware analysis, detection, and classification methods. In Sect. 3, a malware analysis method using bitmap images and entropy graphs is proposed, and experimental results are shown in Sect. 4. In Sect. 5, limitations and future work are discussed. Finally, in Sect. 6, the conclusions of this paper are presented.

2 Related work

There are various malware detection and classification methods, including graph-based methods [7–9], instruction sequence-based methods [10,11], instruction frequency-based methods [12,13], tainted analysis [14], API call monitoring methods [15,16], and behavior-based methods [17] have been proposed to detect and classify malware. Even though there have been many researches on static and dynamic analysis, these analysis methods still have problems with new malware and their variants.

Recently, several visualization techniques have been proposed to compensate or to help malware analysis. These techniques allow human analysts to observe the features of malware visually. Trinius et al. [18] visualized the behavior of malware into *treemap* and *thread graph* by collecting

K. S. Han · J. H. Lim · B. Kang
Department of Electronics and Computer Engineering,
Hanyang University, Seoul, Korea
e-mail: 1hanasun@hanyang.ac.kr

E. G. Im (✉)
Division of Computer Science and Engineering,
Hanyang University, Seoul, Korea
e-mail: imeg@hanyang.ac.kr

information about the API calls and the operations of the performed actions in a sandbox. The treemap shows information such as percentage of API calls and section information, and the thread graph shows the actual chronological behaviors of a malware. Saxe et al. [19] proposed a system to visualize the shared system call sequence relationships. They extracted meaningful system call sequences from system call logs and constructed a Boolean vector representation of the malware binary file corpus, and then, two visualization interfaces have implemented. The first shows map-like visualization of similarity, and the second shows similarities and differences between selected samples.

The static analysis-based visualization techniques also have been proposed. Conti et al. [20] presented an integrated visualizing system that contains many graphical visualization techniques. Their system shows each byte, the presences of bytes, and duplicated sequences of bytes contained within a sample. Because of the overhead of dot plot algorithm, they implemented simplified algorithm by applying these visualization techniques. Anderson et al. [21] visualized the similarity measurement results using all of the available information extracted through the static analysis and dynamic analysis into images named *heatmap*. Nataraj et al. [22] proposed a method to classify malware by using image processing. The proposed method represented executable binary files into gray-scale bitmap images by scanning every bit in binary files by converting each bit value into an image pixel. After generating images, they applied an abstract representation technique for the scene image, i.e., GIST [23–25], to compute texture features and to classify malware. They proved that the binary texture analysis techniques using image processing can classify malware more quickly than existing malware classification methods [26]. However, since the texture analysis method has large computational overheads, the proposed method has problems to process a large number of malware. D. Baysa et al. [27] used two-dimensional matrix to compare malware, but the approach has high computational overheads because 3-gram opcode sequences are used to generate the matrix.

3 Proposed method

3.1 Overview

In this paper, we propose a malware analysis method using visualized images and entropy graphs. The proposed method consists of three steps, as shown in Fig. 1. In Step 1, the "Bitmap Image Converter" receives Windows PE (Portable Executable) binary files as inputs and converts binary files into bitmap images to visualize the binary files. After the bitmap image conversion, in Step 2, the "Entropy Graph Generator" calculates the entropy value of each line of bitmap images and generates entropy graphs based on these values to analyze the similarities of original binary files. In addition, binary files in the Windows PE format are divided into sections for further analysis. In Step 3, the bitmap images and entropy graphs are stored in the database as features of malware, and the features are used to detect and classify malware by calculating similarities of entropy graphs.

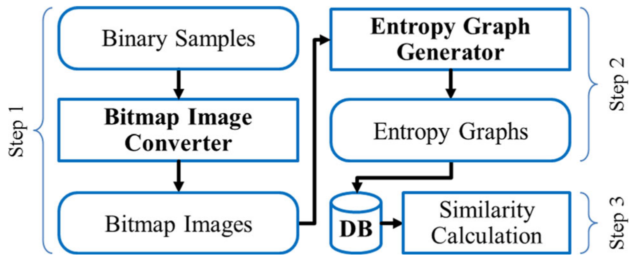


Fig. 1 Visualized analysis method

Note that the bitmap images in our proposed malware analysis method are used as following purposes. First, they allow malware researchers to understand the structures of malware binary files without disassembling and to make a decision for applying of unpacking. Second, the bitmap images can be used to identify packed sections as well as to classify malware families. The purpose of this paper is a malware family classification based on the entropy graph similarity.

3.2 Bitmap image conversion

A technique to represent different files with gray-scale images was presented in [28], and a technique of converting binary files to bitmap images became concrete in [22]. In this paper, we adopted the converting techniques and implemented a "Bitmap Image Converter".

As shown in Fig. 2, the "Bitmap Image Converter", including three modules named (1) section initializer, (2) byte scanner and (3) BMP recorder, converts input binary files into gray-scale bitmap images so that malware researchers can visually analyze the binary files. In order to convert

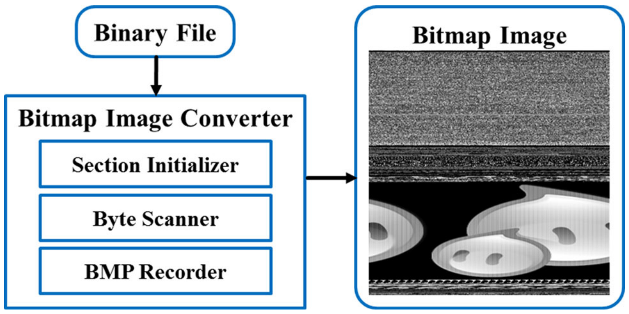


Fig. 2 Bitmap image conversion

binary files into bitmap images, first, the “Section Initializer” extracts information about sections such as the name of the section, the starting offset, and the size of the section respectively from the header of the PE binary file. Second, the “Byte Scanner” scans bytes of the binary file based on the section information, and then, “BMP Recorder” stores the byte values in bitmap pixels. As a result, a binary file can be converted to a gray-scale image.

The converted bitmap images can contain patterns that can be judged as binary features. Since the input binary files in the PE format are used in Windows operating systems, icons of malware can also be extracted from the resource section.

3.3 Entropy graph generation

When binary files are converted into bitmap images, malware researchers can visually compare the bitmap images. However, one of the problems is that similarity calculation is difficult and time-consuming, and analysis automation is even more difficult. To solve these problems, we generated entropy graphs based on the entropy values from the bitmap images so that automated comparison can be possible.

As shown in Fig. 3, the “Entropy Graph Generator” generates entropy graphs based on the bitmap image. That is, the gray-scale bitmap image is used to calculate entropy values, which are used to generate a entropy graph. For this purpose, entropy graph generator calculates those entropy values using color values of each line of the bitmap image and then represents these values as a graph called entropy graph. For example, in case of the image in which the width is 256 pixels, the entropy value of 256 pixels is represented as one pixel of an entropy graph. As a result, an entropy graph for an entire binary file can be generated. An entropy graph represents entropy values in y axis and heights of bitmap images in x axis.

Data of entropy graph is calculated as follows:

– *Entropy*: Entropy is a value that indicates how irregular values appear. If occurrences of all values are same, the entropy value will be the largest. On the contrary, if certain byte values occur with high probabilities, the entropy value will be smaller. Therefore, entropy can be used to compare bitmap images. Equation (1) is used in entropy calculations, where p_i refers to the probability of appearances of a byte value i [29]. The range of values was set to 0 through 255 because byte code values are in the range of 0 through 255.

$$Entropy = -\sum_{i=0}^{255} p_i \times \log_2 p_i \tag{1}$$

3.4 Entropy graph similarity measurement

In this paper, the entropy graphs used for the similarity calculation of the malware binary files are generated using the entire section. Generally, the *.text* section contains executable code. However, some malware binary files do not include the *.text* section, but include the section with different names such as *.CODE*. Therefore, we focused on the entropy graph of the entire section.

After entropy graphs are generated, numerical similarity measurement can be conducted automatically through an

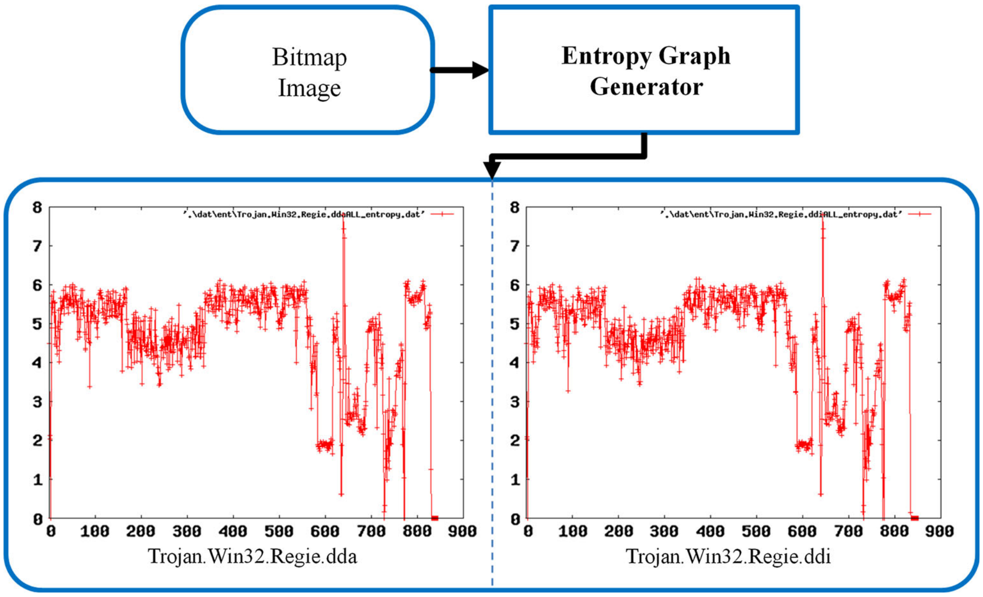
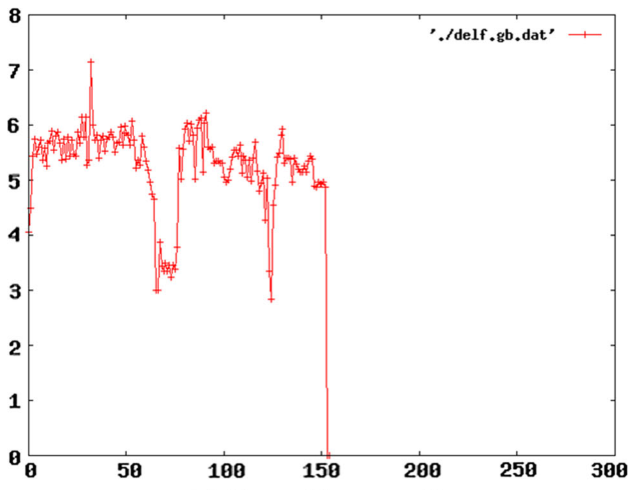
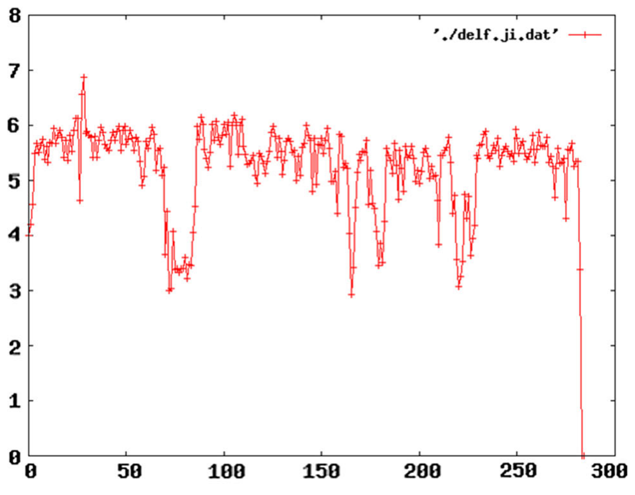

Bitmap Image
Entropy Graph Generator
'.\dat\ent\Trojan.Win32.Regie.ddaALL_entropy.dat'
'.\dat\ent\Trojan.Win32.Regie.ddiALL_entropy.dat'
Trojan.Win32.Regie.dda
Trojan.Win32.Regie.ddi

Fig. 3 Entropy graph generation

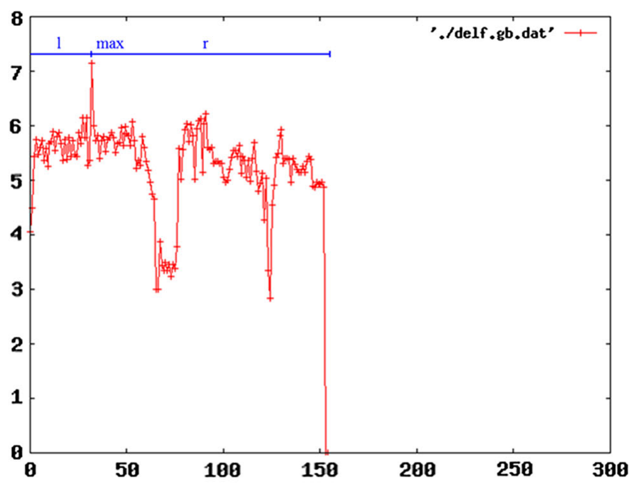


(a) Shorter histogram H

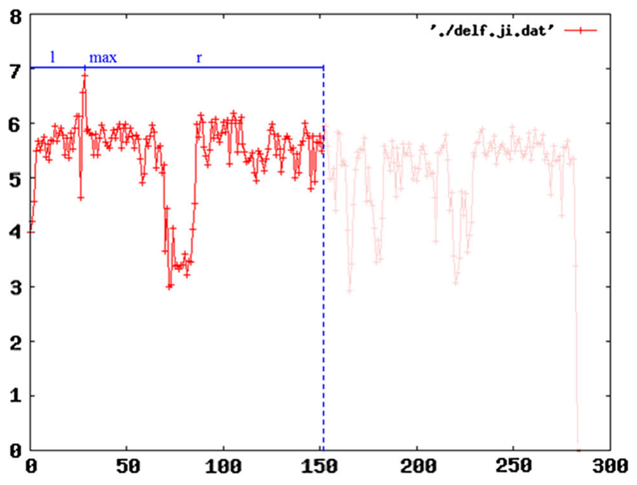


(b) Longer histogram L

Fig. 4 Entropy graphs greatly differ from each other in length



(a) Shorter histogram H



(b) Longer histogram L

Fig. 5 Entropy graphs after solving the problem of different lengths

entropy graph similarity calculation algorithm. To measure the similarity between two entropy graphs, we modified the histogram similarity measuring method proposed by Strelkov et al. [30]. To measure the graph similarities, the following two values, k_1 and k_2, are used.

– $\mathbf{k}_1$: k_1 is a value calculated using the cumulated sum of differences of entropy values in the same x-position. Therefore, k_1 indicates the similarity of general forms of two entropy graphs. When k_1 is calculated, the computation that must be performed first is to align the x-axes (the heights of bitmap images) of the two entropy graphs. For instance, as shown in Fig. 4, if the shorter entropy graph H is included in the other entropy graph L, the maximum entropy value, i.e., max, found in the shorter entropy graph H, the corresponding values of the left side length l and the right side length r of max are used to shorten the longer entropy graph L, as shown in Fig. 5, to equalize the x-axes before k_1 is calculated.

Equation (2) is used to calculate k_1 after the lengths of the x-axes of two entropy graphs are equalized, where $s[H, L]$ refers to the sum of distances between the two entropy graphs, and $\bar{s}$ is the area of the entropy graph.

$$k_1[H, L] = \exp\left(-\frac{s[H, L]}{\bar{s}}\right) \tag{2}$$

where

$$s[H, L] = \int_x |H(x) - L(x)|\, dx \tag{3}$$

$$\bar{s} = Average(H) \times Length \text{ of } y \text{ axis} \tag{4}$$

For the two entropy graphs H and L, distances in each x axis point are calculated and accumulated. After calculating the accumulated value for all the points, the sum is divided by $\bar{s}$, and k_1 is calculated through the function exp(). If two entropy graphs are perfectly the same, no distance will exist, and thus, the value of k_1 will become 1. Since the range of the values of $s[H, L]$ is from 0 to infinity, when the input value, $s[H, L]$, is 0, k_1 is the maximum value of 1, and as the input value increases, k_1 converges toward 0 on the contrary.

– $\mathbf{k_2}$: k_2 is a value calculated using the distance between local maxima of entropy graphs. In this paper, local maxima with larger variations in each entropy graph would be reflected more on k_2. Equation (5) is used to calculate k_2. Equations (6)–(9) are also used to calculate individual elements.

$$k_2[H, L] = \sum_i u_i[H]c_i[H, L] \tag{5}$$

where

$$u_i = \frac{\left|H^{(2)}(x_i)\right| \times l_i}{\sum_{i=1}^{n(H)} \left|H^{(2)}(x_i)\right| \times l_i} \tag{6}$$

$$c_{i,j} = c^x_{i,j} \times c^y_{i,j} \tag{7}$$

$$c^x_{i,j} = -\exp\left(\frac{\Delta x_{i,j}}{\delta x}\right)^2$$

$$\Delta x_{i,j} = x_i - \tilde{x}_j \tag{8}$$

$$c^y_{i,j} = -\exp\left(\frac{\Delta y_{i,j}}{\delta y}\right)^2$$

$$\Delta y_{i,j} = H(x_i) - L(\tilde{x}_j) \tag{9}$$

u_i refers to the ratios of reflection of local maxima, where c_i is calculated using the distance between the ith local maximum and the local maximum closest to the ith local maximum; l_i refers to the distance between a straight line formed by two points on both sides of a local maximum where variations become between 0 and the local maximum. For l_i to have a high value, the $H(x)$ values of the two points should be maximally similar to each other, and the $H(x)$ value of the local maximum should be larger than the $H(x)$ values of these two points. The larger value of the second derivative at $H(x)$ makes the sharper shape of the local maxima. Therefore, the sharper and higher shape of local maxima, the larger the

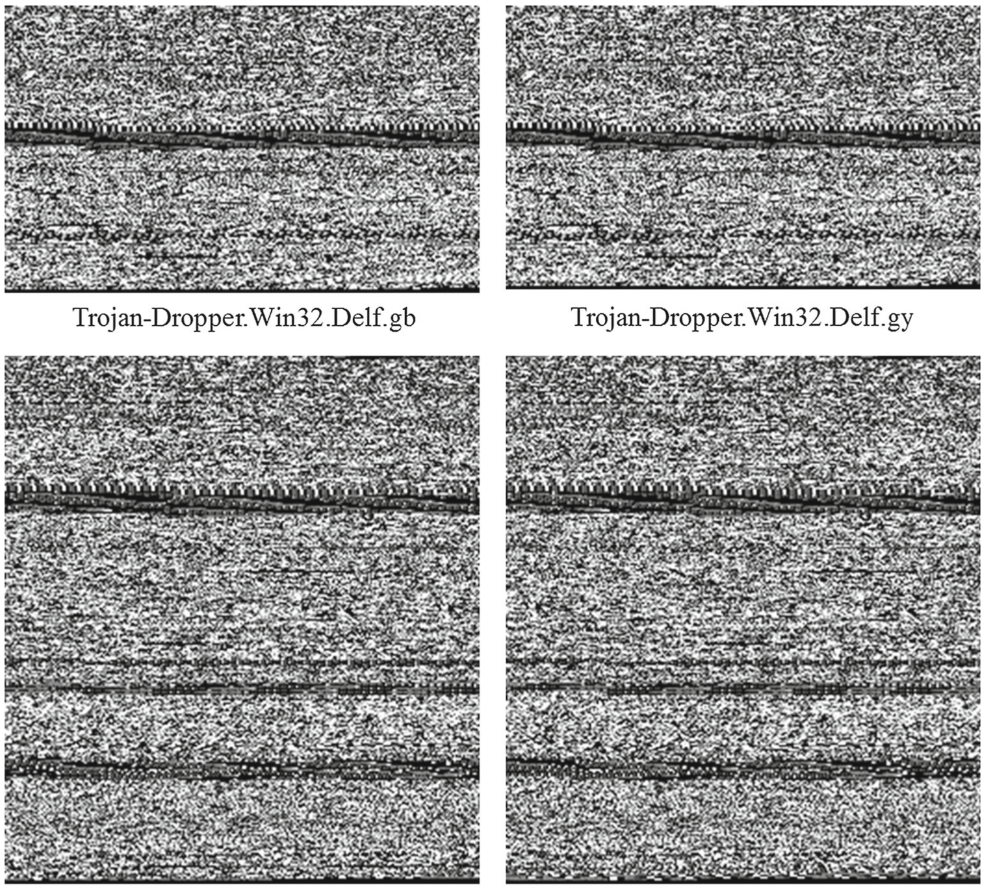

Trojan-Dropper.Win32.Delf.gb
Trojan-Dropper.Win32.Delf.gy
Trojan-Dropper.Win32.Delf.ji
Trojan-Dropper.Win32.Delf.xf

Fig. 6 Gray-scale images of Delf family

weighted value given to the relevant local maxima. $n(H)$ refers to the number of local maxima on entropy graph H.

The value of c_i is a double of the value obtained by function exp(), and the resultant value ranges from 0 to 1. The result of the multiplication of the two values also ranges from 0 to 1. If all resultant values of c_i are 1, the sum of u_i will also be 1, and thus, k_2 will be 1. As the values of c_i converge on 0, k_2 also converges on 0. Therefore, k_2 ranges from 0 to 1. Detailed explanation of k_2 can be found in [30].

When k_1 and k_2 have been calculated, these two values are added up to calculate the final similarities of the entropy graphs. The ratios of reflection of k_1 and k_2 are determined to use the sum of the values of k_1 and k_2 in the measurement of the similarities between entropy graphs. Equation (10) is used in the measurement of entropy graph similarities using k_1 and k_2, where t_1 and t_2 are weighted values given to the values of k_1 and k_2, and the sum of them is 1. The values of t_1 and t_2 used in this paper are 0.7 and 0.3, respectively.

$$S = t_1 \times k_1 + t_2 \times k_2 \tag{10}$$

4 Experimental results

4.1 Experimental environment and data

The proposed method was implemented in the C language using Visual Studio 2010. Total 24 benign binaries and 27 malware binary files from 8 families were tested in the first experiments. Malware binary files used in the experiments were collected from VX Heavens [31]. The collected malware binary files were Backdoors, Trojans, Viruses, and Worms that are executable files in the Windows operating system. We used malware file names assigned by Kaspersky [32]. In order to generate entropy graphs, we used Gnuplot [33], which can draw entropy graphs only with coordinate data and simple commands. After examining the possibilities for malware classification and determining a threshold to minimize false-positives through the first experiments using small data, then we experimented with large-scale malware data.

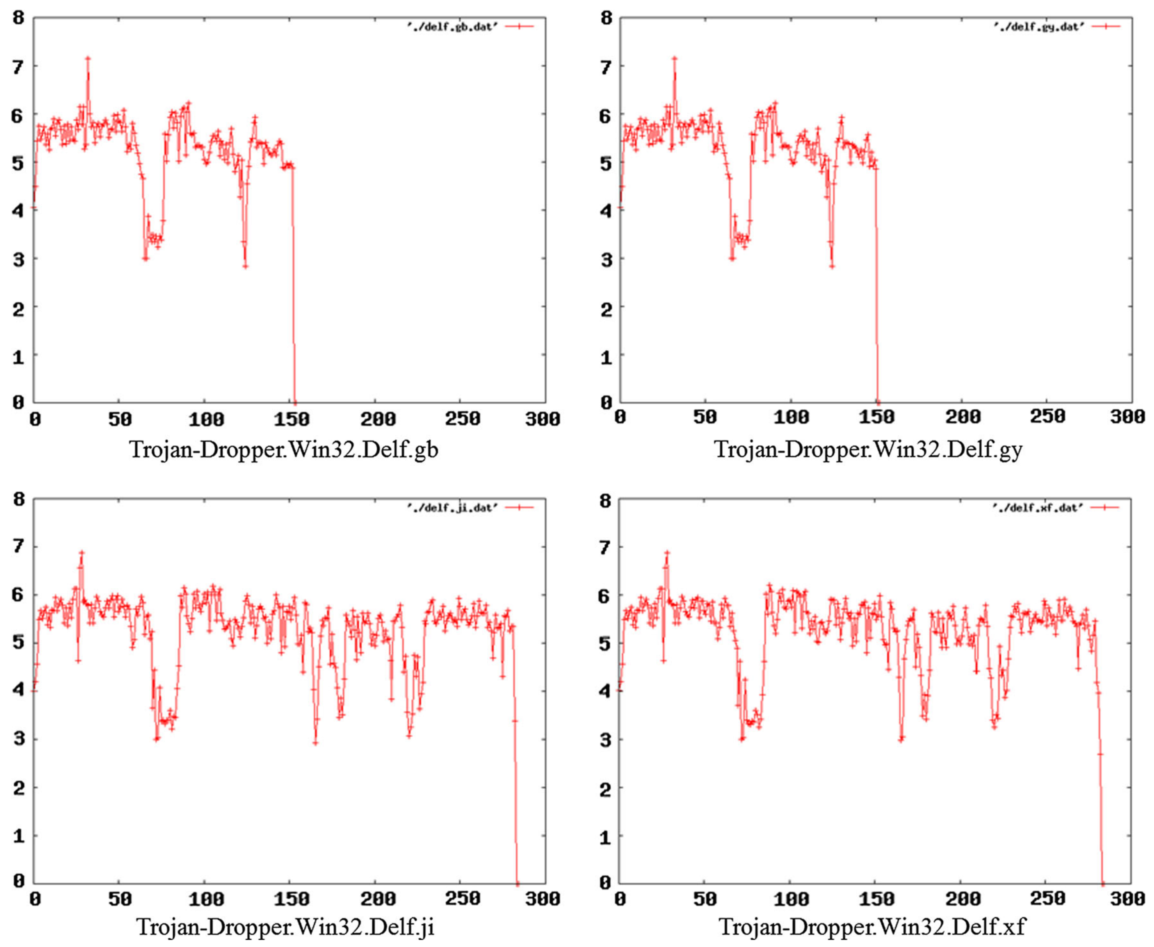


Fig. 7 Entropy graphs of Delf family

4.2 Results of experiments conducted with same malware family variants

In this subsection, we will explain experimental results of malware variants, which belong to the same family.

– *Trojan-Dropper.Win32.Delf:* Figure 6 shows the gray-scale images of variants of the Trojan-Dropper.Win32.Delf family, and Fig. 7 shows corresponding entropy graphs based on entropy values. In Fig. 6, it can be seen that a particular pattern can be identified despite the heights of bitmap images being different, since the sizes of individual binary files are different. It can be also seen from the entropy graphs in Fig. 7 that the patterns of the entropy values are similar to one another. In addition, as mentioned in Sect. 3.4, since the similarities are calculated with reference to the maximum value of the graph, their similarities have high values. The results of the similarities for the entropy graphs of Delf family are shown in Table 1.

– *Virus.Win32.Evol:* Figure 8 shows gray-scale images and entropy graphs of the entire section from the Virus.Win32.
Evol family. Because the sizes of malware binary files are small, the heights of the bitmap images are low and the maximum values of the *x* axis of the entropy graphs

Table 1 Entropy graph similarities of Delf family

Trojan-Dropper.Win32.Delf				
	gb	gy	ji	xf
gb	1	0.920	0.754	0.759
gy	0.920	1	0.760	0.763
ji	0.754	0.760	1	0.962
xf	0.759	0.763	0.962	1

Table 2 Entropy graph similarities of Evol family

Virus.Win32.Evol			
	a	b	c
a	1	0.938	0.943
b	0.938	1	0.991
c	0.943	0.991	1

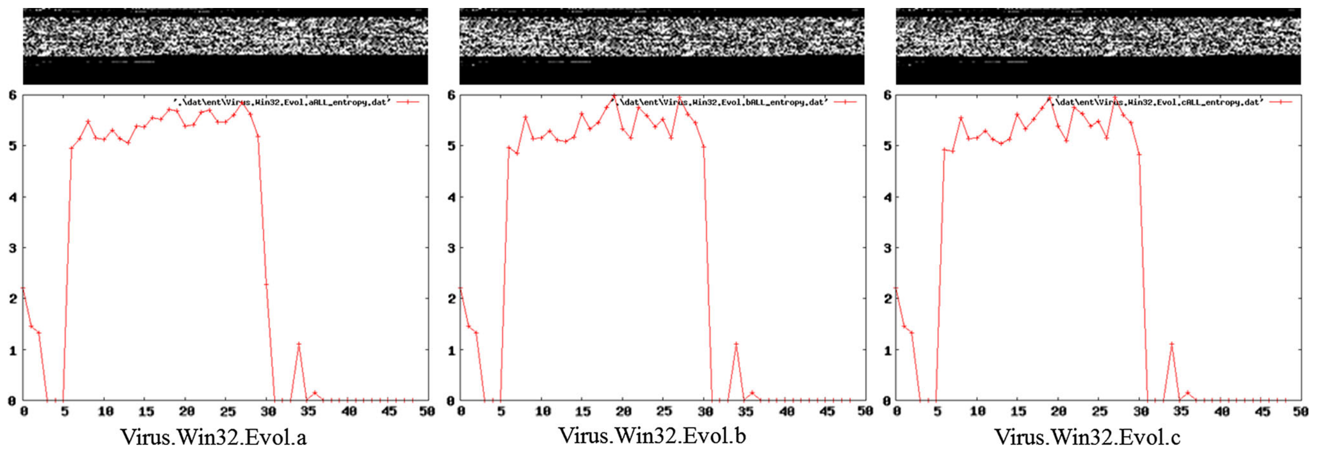


Fig. 8 Gray-scale images and entropy graphs of Evol family

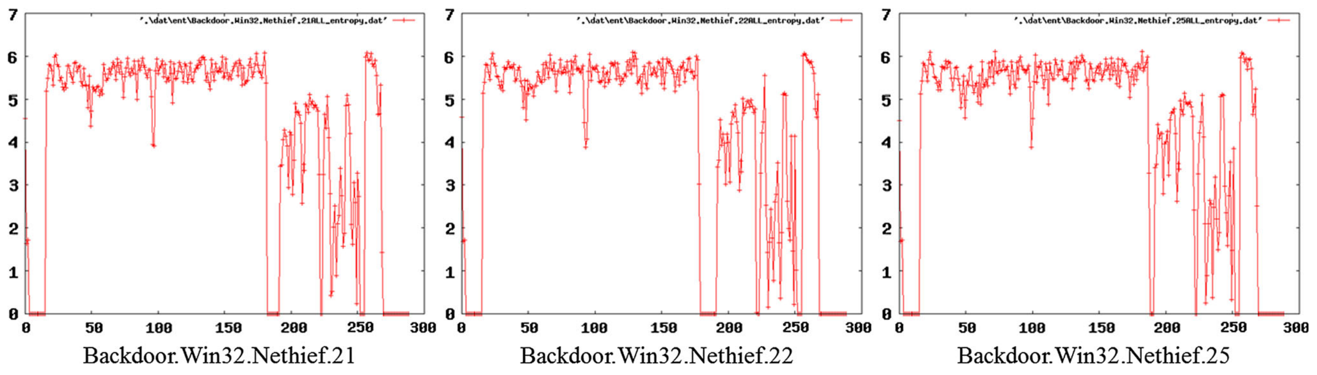


Fig. 9 Entropy graphs of Nethief family

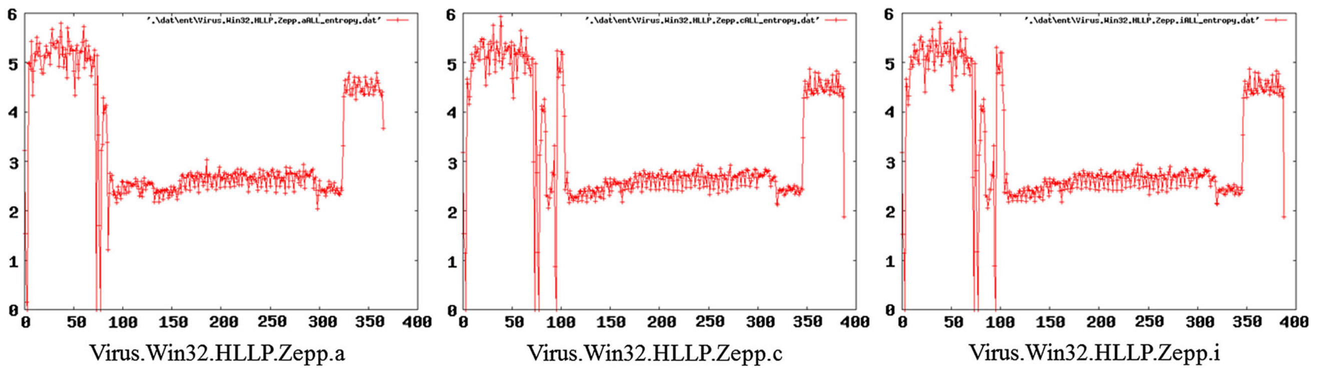


Fig. 10 Entropy graphs of Zepp family

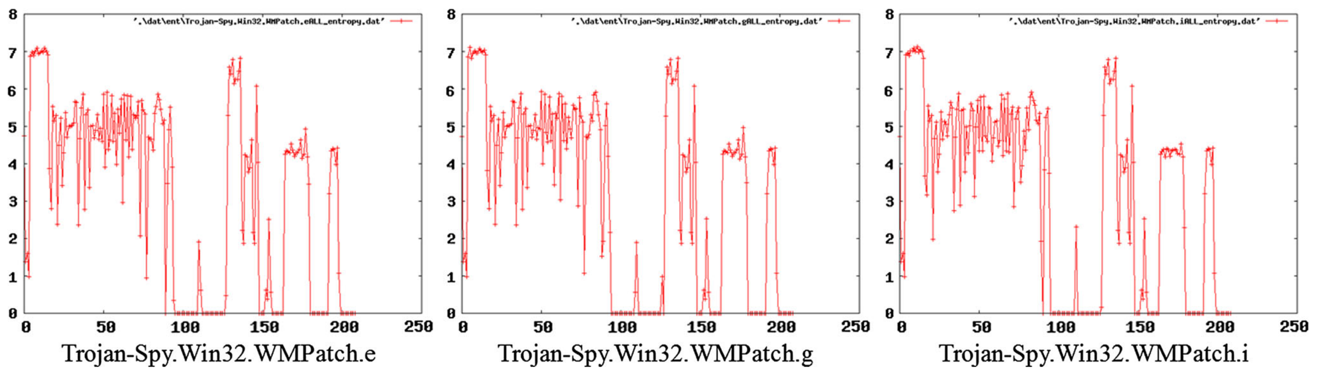


Fig. 11 Entropy graphs of WMPatch family

Table 3 Entropy graph similarities of Nethief family

Backdoor.Win32.Nethief			
	21	22	25
21	1	0.889	0.951
22	0.986	1	0.949
25	0.951	0.949	1

Table 4 Entropy graph similarities of Zepp family

Virus.Win32.HLLP.Zepp			
	a	c	i
a	1	0.889	0.880
c	0.889	1	0.982
i	0.880	0.982	1

are small too. As shown in Table 2, the similarities among the entropy graphs are high because the entropy graphs are quite similar.

– *Other Malware Families:* Similarly, Figs. 9, 10, and 11 show the entropy graphs of Backdoor.Win32.Nethief, Virus.Win32.HLLP.Zepp and Trojan-Spy.Win32. WM Patch families. Also, Tables 3, 4, and 5 show entropy graph similarities of individual families. The average similarity of each family was greater than 0.9 because of similar sizes of the binary files.

Table 5 Entropy graph similarities of WMPatch family

Trojan-Spy.Win32.WMPatch			
	e	g	i
e	1	0.985	0.983
g	0.985	1	0.981
i	0.983	0.981	1

4.3 Classification threshold

Figure 12 presents similarities of five malware families and benign files. The left side of the chart has benign files and the right side has malware binary files. The similarities between variants of the same malware family have high values over the average 0.7 compared with those values for benign and other malware binary files. Therefore, the malware family classification can be possible by calculating the similarities among entropy graphs of unknown binary files, those of benign and malware binaries.

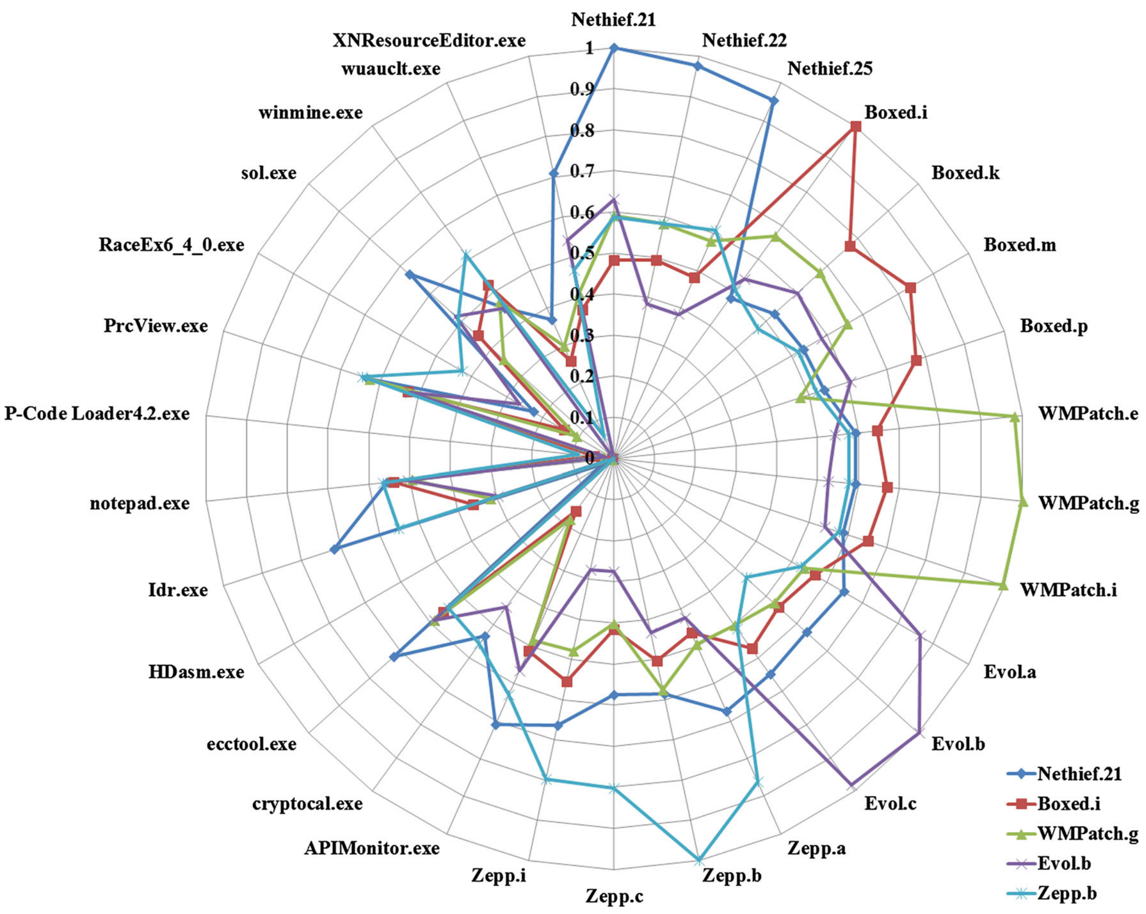


Fig. 12 The result of similarities among benign and malwaew binary files

The false-positive rates and false-negative rates according to the different threshold values are shown in Fig. 13. The false-positive rate will be increased if benign files are detected as malware [34]. On the other hand, if malicious binary files are detected as benign, then the false-negative rate will be increased. When the threshold value for classification is increased, the false-positive rate is reduced, but the false-negative rate is increased. The threshold value of 0.75 is the best combination of false-positive rate and false-negative rate in Fig. 13. To classify new or unknown malware, we can use 0.75 as a threshold, and our proposed method will have about 3 % false-negative rate and 3.5 % false-positive rate.

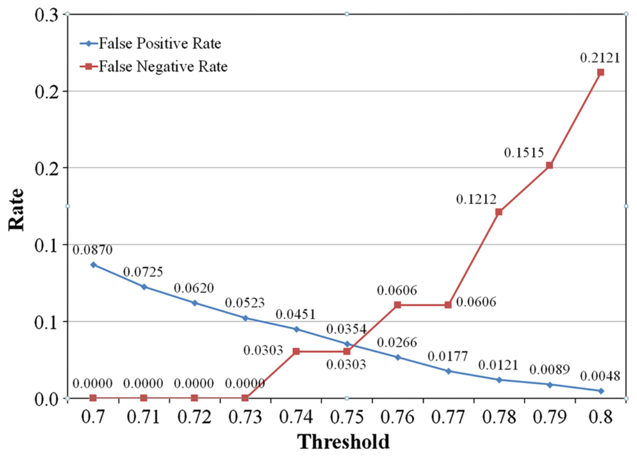


Fig. 13 The changes of false-positive rate and false-negative rate according to the threshold

4.4 Results of experiments using large-scale data

In the previous experiments, we examined the possibilities of the malware classification based on the similarities of the entropy graphs using small data. In addition, we discovered the classification threshold which can minimize both false-positives and false-negatives.

This section shows experimental results with large-scale data. As shown in Table 6, we used total 1,000 malware binary files from 50 families. Similarities are calculated by comparing entropy graphs, and Fig. 14 shows the average similarities among individual malware families. The figure shows

Table 6 Large-scale data of malware binary files

Type	Family (# of Variants)	
Backdoor	Obana (13)	
Worm	Chiviper (16)	Downloader (72)
Email-Worm	Joleee (16)	LoveLetter (13)
	Mydoom (14)	Warezov (12)
Net-Worm	Allaple (15)	Morto (17)
Rootkit	HareBot (14)	Hodprot (25
	Small (40)	Zybr (19)
Trojan	Bepiv (10)	Favadd (16)
	Fidgen (15)	Krament (15)
	Lampa (23)	Llac (16)
	Migotrup (14)	Pakes (10)
	Refroso (13)	Regrun (17)
	Remex (15)	Rozena (14)
	Small (58)	Starter (18)
	Wigon (94)	
Trojan-Banker	Banker (14)	
Trojan-Downloader	Deliver (11)	Helminthos (13)
	ILovlan (20)	Myxa (10)
	Pher (10)	Plosa (7)
	Winlagons (15)	
Trojan-FakeAV	AntiSpyware (16)	WinAntiVirus (8
Trojan-PSW	Element (10)	FireThief (15)
	MailRu (16)	QQPass (61)
	Rebnip (15)	YahuPass (13)
Trojan-Ransom	Birele (18)	BlackDeath (6)
	Cidox (7)	Hexzone (15)
Trojan-Spy	Ardamax (40)	Flux (26)
Total	50 (1000)	

a clear distinction between similarities of the same families and those of other families. With the threshold value of 0.75, our family classification method had the accuracy of 0.979, the accuracy result is similar to that of the proposed method in [22]. Twenty-one malware binary files among 1,000 were

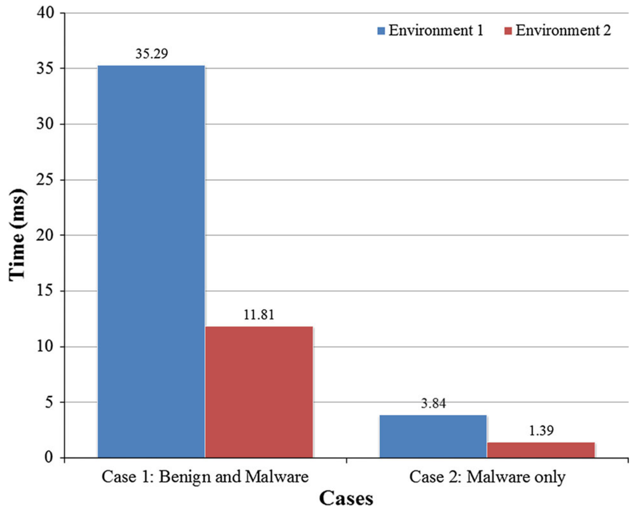


Fig. 15 Average similarity calculation times for two cases

Table 7 Two environments for performance test

	Environment 1 (old PC)	Environment 2 (new PC)
CPU	Intel Core2Duo E8400 3.0 GHz (2 cores)	Intel i7-4770 3.4 GHz (4 cores)
Memory	DDR2 2 GB	DDR3 8GB
OS	Windows 7 32 bit	Windows 7 64 bit

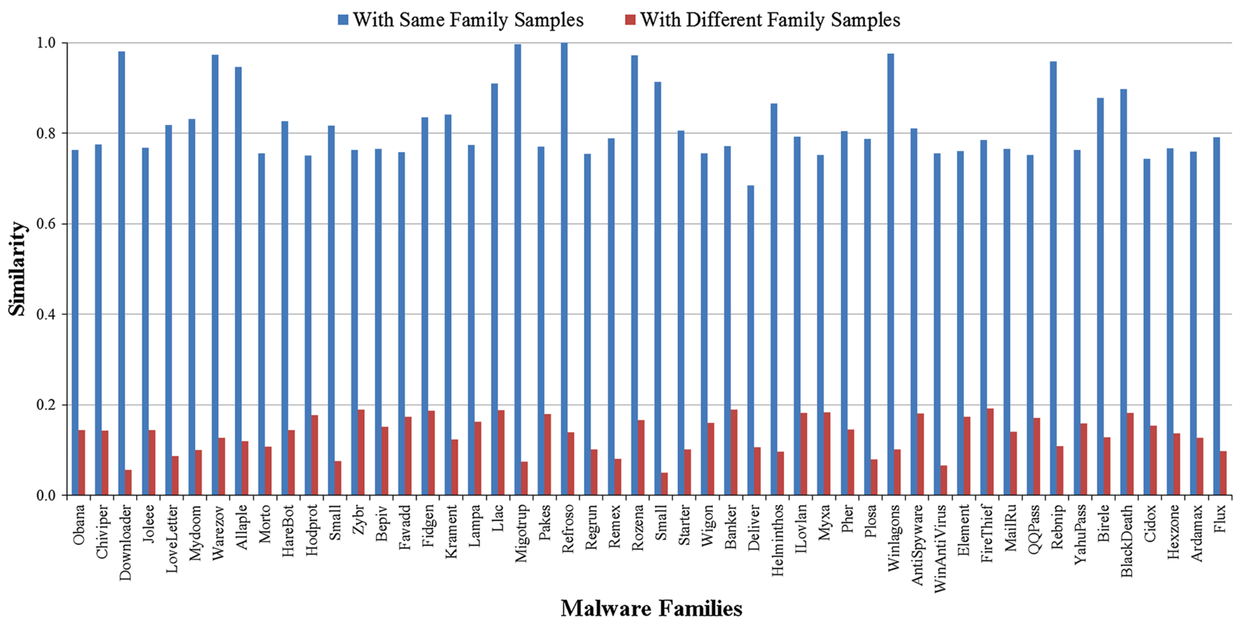


Fig. 14 Similarity calculation results for individual malware families

classified into the other families. Incorrectly classified malware binary files belong to Deliver, QQPass, and Wigon.

4.5 Computational overheads

Our entropy graph-based method has less computational overheads than texture analysis methods, such as GIST used in [22]. In order to evaluate the computational overheads of our proposed method, we measured the similarity calculation time of two cases on two general PC environments as shown in Table 7, and compared our method with the GIST-based method.

Figure 15 shows the results of similarity calculation time for each case. In the first case, average times to calculate the

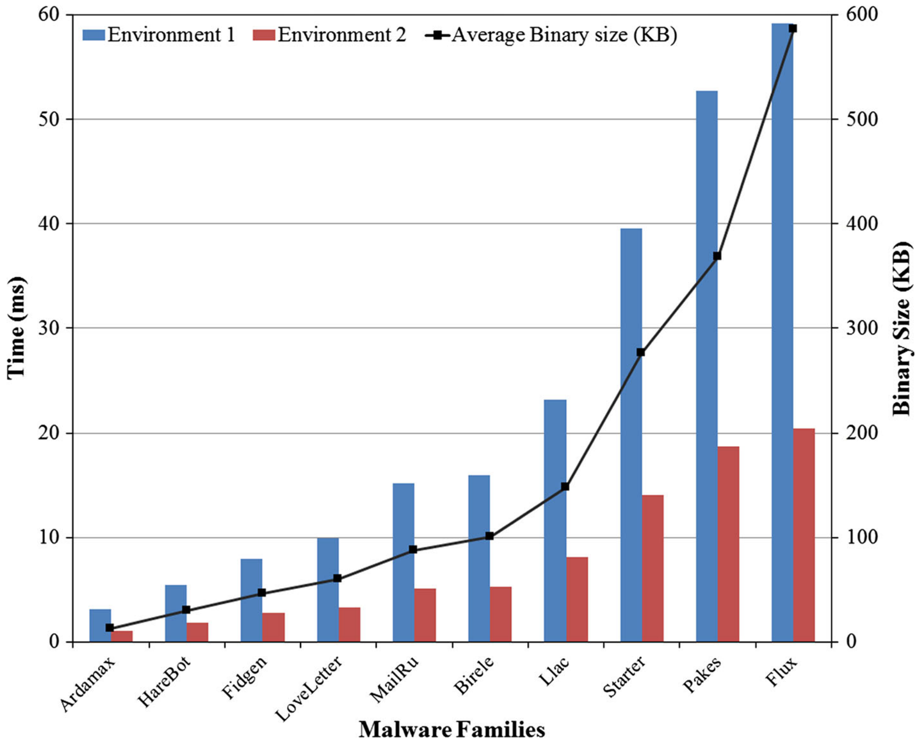


Fig. 16 Similarity calculation times according to different binary sizes

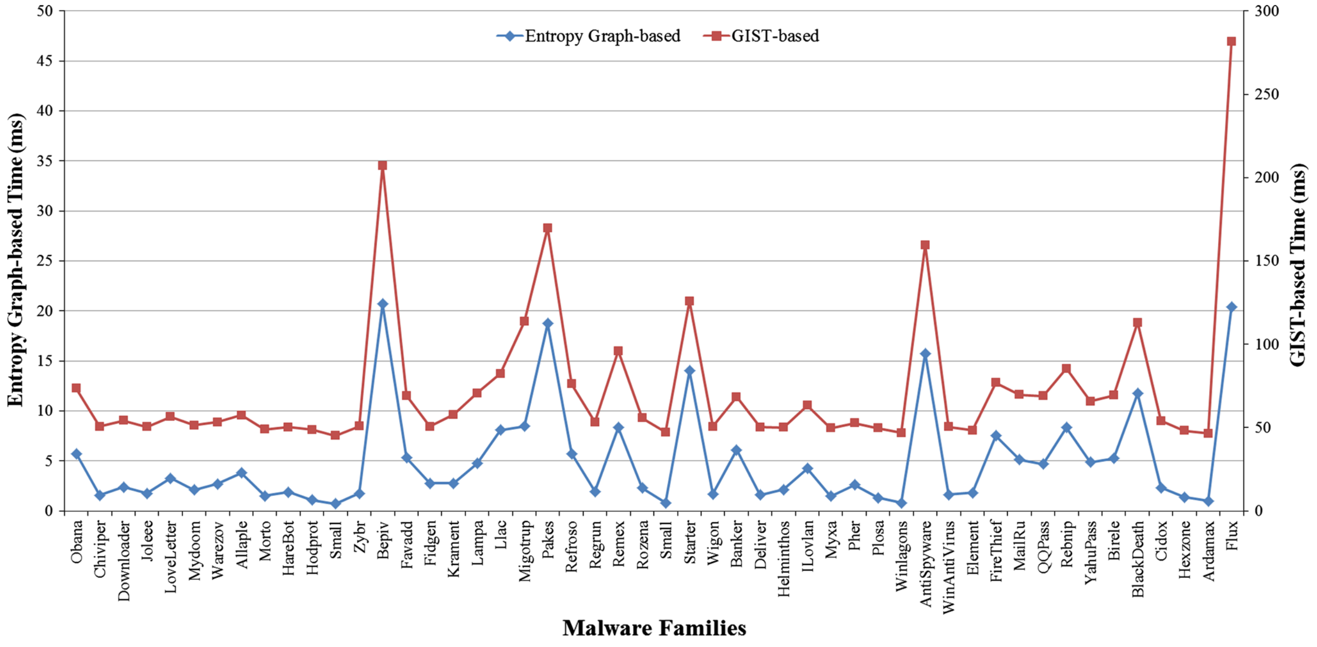


Fig. 17 Comparison of the similarity calculation time with the method of [22]

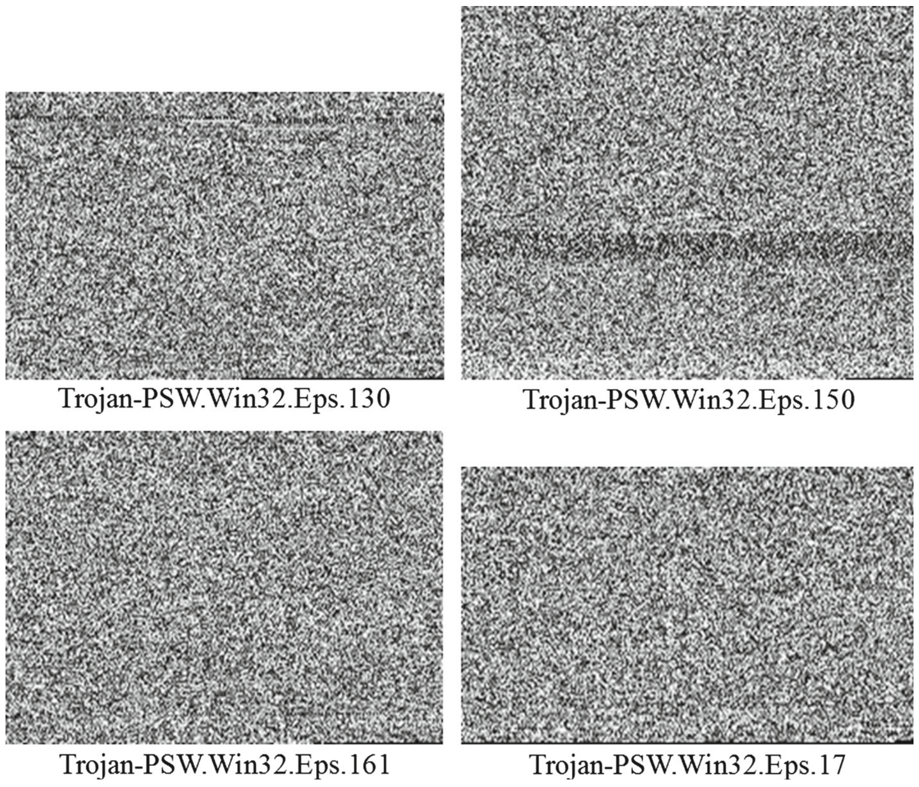


Fig. 18 Gray-scale images of the .text sections of EPS family

similarities of all the benign and malware binary files were 35.29 ms in environment 1 and 11.81 ms in environment 2, respectively. In the second case, average times to calculate the similarities among malware binary files only were 3.84 and 1.39 ms in each environment, respectively.

The similarity calculation time depends on the size of the binary file. Figure 16 shows the average similarity calculation time and binary sizes of malware, and the graph shows that similarity calculation time is proportional to the binary file size.

Even if the differences exist in the average similarity calculation times according to the sizes of binary files and the environments, the results show that our proposed method is faster than the GIST-based method proposed in [22], which shows the average computing time was 54 ms. Actually, as shown in Fig. 17, two methods had been directly compared in our environments, and the results show that our proposed method was still faster than GIST-based texture analysis.

5 Limitation and future work

Figure 18 shows the gray-scale images of the malware Trojan-PSW.Win32.EPS family. Since no particular pattern can be identified in Fig. 18, the entropy values of binaries can be very high. That is, when binary files are packed, packed sections usually have high entropy values. Figure 19 shows entropy graphs of packed binaries. The entropy values of packed binaries maintain values close to 7 in almost all sections. Therefore, if the binary files have a similar pattern like entropy graph in Fig. 19, those binary files can be judged to be packed, and the packed malware binaries are difficult to classify through the similarity calculation using the entropy graphs.

In our proposed method, the malware binary files in which the packing techniques are applied are beyond the scope of this paper. Therefore, these malware binary files have to be unpacked before our proposed malware analysis method is applied. Note that the bitmap images can be used to detect packed malware binary files.

As future studies, in order to analyze the malware binary files in which packing techniques are applied, our proposed method can be extended to instruction-level analysis with the help of dynamic analysis.

6 Conclusions

This paper proposed a malware family classification method using visualized images and entropy graphs. The contributions of the paper are the following:

- We proposed a malware visualization method that converts binary files into gray-scale bitmap images and generates entropy graphs.
- We also proposed a method to calculate similarities of malware using visualized images and to classify malware families.

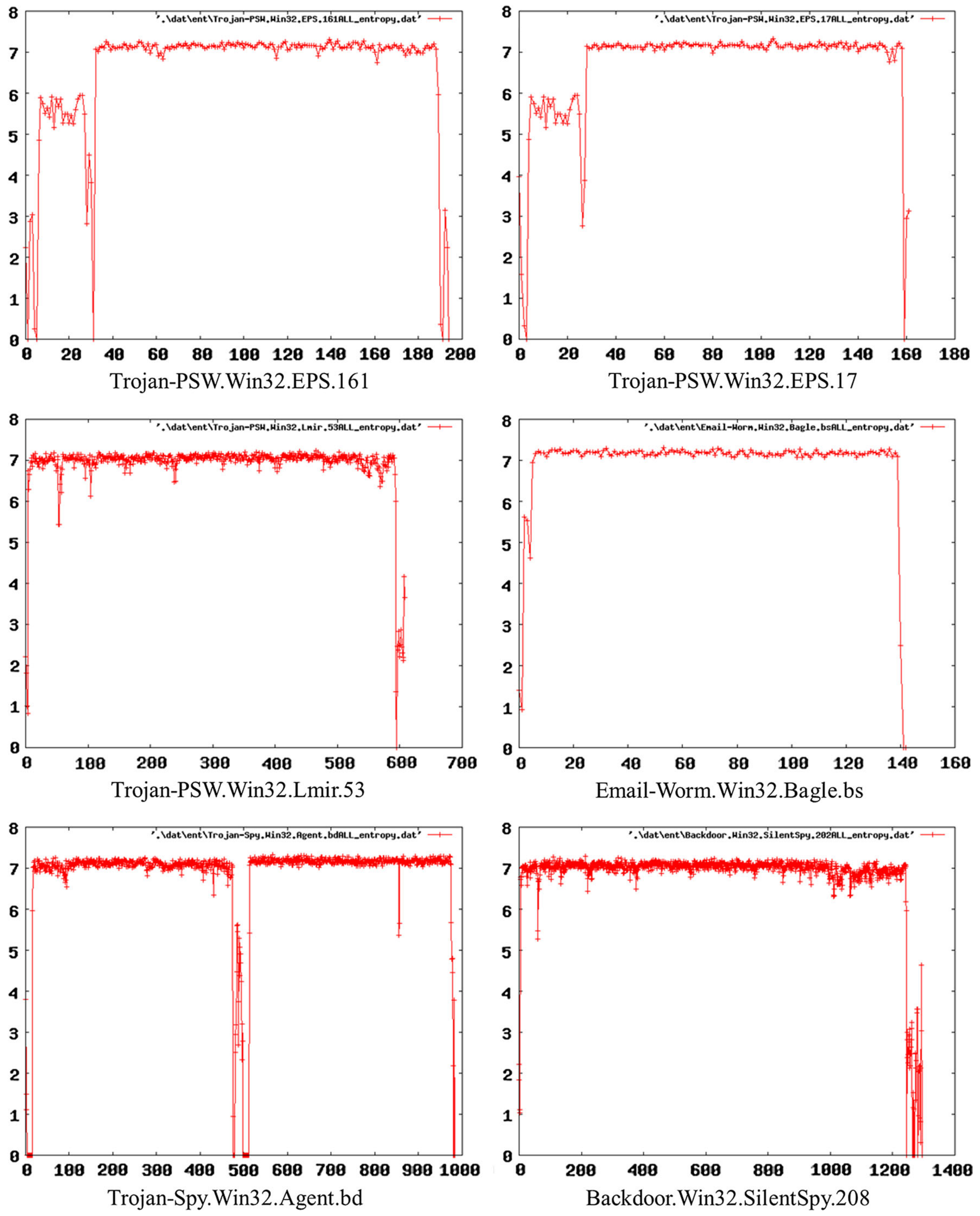


Fig. 19 Entropy graphs of packed binaries

– Experimental results showed that our proposed method can classify malware families with a small false-positive/false-negative rate.

Our proposed method can be easily automated and can be used to preprocess a large number of new or unknown malware binary files. As a result, our method can reduce the number of malware binary files need to be analyzed in dynamic and static analysis.

Acknowledgments This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1022) supervised by the NIPA (National IT Industry Promotion Agency).

References

1. Christodorescu, M., Jha, S.: Testing malware detectors. ACM SIGSOFT Softw. Eng. Notes **29**(4), 34–44 (2004)
2. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual 2007, pp. 421–430. IEEE
3. Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using cwsandbox. IEEE Secur. Priv. **5**(2), 32–39 (2007)
4. Jang, J., Brumley, D., Venkataraman, S.: Bitshred: feature hashing malware for scalable triage and semantic analysis. In: Proceedings of the 18th ACM Conference on Computer and Communications Security 2011, pp. 309–320. ACM
5. Dinaburg, A., Royal, P., Sharif, M., Lee, W.: Ether: malware analysis via hardware virtualization extensions. In: Proceedings of the 15th ACM Conference on Computer and Communications Security 2008, pp. 51–62. ACM
6. Christodorescu, M., Jha, S., Seshia, S.A., Song, D., Bryant, R.E.: Semantics-aware malware detection. In: Security and Privacy, 2005 IEEE Symposium on 2005, pp. 32–46. IEEE
7. Cesare, S., Xiang, Y.: A fast flowgraph based classification system for packed and polymorphic malware on the endhost. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on 2010, pp. 721–728. IEEE
8. Chowdhury, G.: Introduction to Modern Information Retrieval. Facet publishing (2010)
9. Shang, S., Zheng, N., Xu, J., Xu, M., Zhang, H.: Detecting malware variants via function-call graph similarity. In: Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on 2010, pp. 113–120. IEEE
10. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: Detection of new malicious code using n-grams signatures. In: Proceedings of Second Annual Conference on Privacy, Security and Trust 2004, pp. 193–196
11. Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C., Bringas, P.G.: Idea: Opcode-sequence-based malware detection. In: Engineering Secure Software and Systems. pp. 35–43. Springer, Berlin (2010)
12. Bilar, D.: Opcodes as predictor for malware. Int. J. Electron. Secur. Digit. Forensics **1**(2), 156–168 (2007)
13. Han, K.S., Kim, S.-R., Im, E.G.: Instruction frequency-based malware classification method. INFORMATION Int. Interdiscip. J. **15**(7), 2973–2984 (2012)
14. Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D.: Dynamic spyware analysis. In: Usenix Annual Technical Conference 2007
15. Nair, V.P., Jain, H., Golecha, Y.K., Gaur, M.S., Laxmi, V.: MEDUSA: MEtamorphic malware dynamic analysis using signature from API. In: Proceedings of the 3rd International Conference on Security of Information and Networks 2010, pp. 263–269. ACM
16. Miao, Q.-G., Wang, Y., Cao, Y., Zhang, X.-G., Liu, Z.-L.: APICapture-a tool for monitoring the behavior of malware. In: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on 2010, pp. V4–390-V394-394. IEEE
17. Fredrikson, M., Jha, S., Christodorescu, M., Sailer, R., Yan, X.: Synthesizing near-optimal malware specifications from suspicious behaviors. In: Security and Privacy (SP), 2010 IEEE Symposium on 2010, pp. 45–60. IEEE
18. Trinius, P., Holz, T., Gobel, J., Freiling, F.C.: Visual analysis of malware behavior using treemaps and thread graphs. In: Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on 2009, pp. 33–38. IEEE
19. Saxe, J., Mentis, D., Greamo, C.: Visualization of shared system call sequence relationships in large malware corpora. In: Proceedings of the Ninth International Symposium on Visualization for Cyber, Security 2012, pp. 33–40. ACM
20. Conti, G., Dean, E., Sinda, M., Sangster, B.: Visual Reverse engineering of binary and data files. In: Visualization for Computer Security, pp. 1–17. Springer, Berlin (2008)
21. Anderson, B., Storlie, C., Lane, T.: Improving malware classification: bridging the static/dynamic gap. In: Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence 2012, pp. 3–14. ACM
22. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber, Security 2011, p. 4. ACM
23. Torralba, A., Murphy, K.P., Freeman, W.T., Rubin, M.A.: Context-based vision system for place and object recognition. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on 2003, pp. 273–280. IEEE
24. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vis. **42**(3), 145–175 (2001)
25. Siagian, C., Itti, L.: Rapid biologically-inspired scene classification using features shared with visual attention. IEEE Trans. Pattern Anal. Mach. Intell. **29**(2), 300–312 (2007)
26. Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence 2011, pp. 21–30. ACM
27. Baysa, D., Low, R.M., Stamp, M.: Structural entropy and metamorphic malware. J. Comput. Virol. Hack. Tech. 9, 179–192 (2013)
28. Conti, G., Bratus, S., Shubina, A., Lichtenberg, A., Ragsdale, R., Perez-Alemany, R., Sangster, B., Supan, M.: A Visual Study of Primitive Binary Fragment Types. White Paper, Black Hat USA (2010)
29. Kapur, J., Sahoo, P.K., Wong, A.: A new method for gray-level picture thresholding using the entropy of the histogram. Comput. Vis. Gr. Image Process. **29**(3), 273–285 (1985)
30. Strelkov, V.: A new similarity measure for histogram comparison and its application in time series analysis. Pattern Recognit. Lett. **29**(13), 1768–1774 (2008)
31. VxHeaven. http://vx.netlux.org/index.html
32. Kaspersky Lab. http://www.kaspersky.com
33. Gnuplot. http://www.gnuplot.info
34. Karampatziakis, N., Stokes, J.W., Thomas, A., Marinescu, M.: Using file relationships in malware classification. In: Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 1–20. Springer, Berlin (2013)