Malware analysis method using visualization of binary files

Conference Paper · October 2013

DOI: 10.1145/2513228.2513294

3 authors, including:

Eul Gyu Im
Hanyang University
117 PUBLICATIONS **670** CITATIONS

SEE PROFILE

Malware Analysis Method using Visualization of Binary Files

KyoungSoo Han
Dept. of Computer and Software,
Hanyang University,
Seoul, Korea
+82-2-2220-2381
1hanasun@hanyang.ac.kr

Jae Hyun Lim
Dept. of Computer and Software,
Hanyang University,
Seoul, Korea
+82-2-2220-2381
zailina@hanyang.ac.kr

Eul Gyu Im
Div. of Computer Science and Engineering,
Hanyang University,
Seoul, Korea
+82-2-2220-4321
imeg@hanyang.ac.kr

ABSTRACT

Malware authors have been generating and disseminating malware variants through various ways, such as reusing modules or using automated malware generation tools. With the help of the malware generation techniques, the number of malware keeps increasing every year. Therefore, new malware analysis techniques are needed to reduce malware analysis overheads. Recently several malware visualization methods were proposed to help malware analysts. In this paper, we proposed a novel method to visually analyze malware by transforming malware binary information into image matrices. Our experimental results show that the image matrices of malware can effectively classify malware families.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security Protection – *Invasive software.*
Security and privacy~Malware and its mitigation

General Terms

Security

Keywords

Malware analysis, malware visualization, malware similarity, malware detection

1. INTRODUCTION

The number of malware found on the Internet continues to increase because malware can be generated with various automated tools and reused modules. Because some modules for malicious behaviors are reused in malware variants, malware variants of the same family have similar binary patterns, and these patterns can be used to detect malware and to classify malware families.

Most antivirus programs focus on malware signatures, i.e string patterns, to detect malware [1]. However, malware variants can avoid these signature-based detection methods by applying diverse detection avoidance techniques [2,3]. Consequently, malware analysts and researchers have been studying diverse analysis techniques in order to deal with malware variants. Since the number of malware increases every year, new malware analysis techniques are needed to reduce burdens of malware analysts. One of new ways of malware analysis is to use visualization techniques.

In this paper, we propose a novel method of visually analyzing malware using the malware binary information to quickly identify, detect, and classify malware and malware families. The proposed method generates RGB colored pixels on image matrices using the malware binary information extracted through static analysis. Using these malware image matrices, similarities are calculated among different malware. Experimental results show that the image matrices of malware can effectively classify malware families. The proposed visualization technique can be easily automated and used to analyze a large number of malware.

This paper is composed as follows. In section 2, malware analysis-related studies are described. In section 3, malware analysis methods using visualized binary information and similarity calculating methods are proposed and experimental results are presented in section 4. Finally, in section 5, conclusions and future directions are provided.

2. RELATED WORK

To detect and classify malware, various static analysis techniques were proposed so far, including control flow graph analysis [4,5], function call graph analysis [6], byte level analysis [7], instruction-based analysis [8,9,10], and similarity-based analysis [11,12]. Even though there are many static analysis techniques available, new techniques that can complement existing techniques are still needed to improve malware analysis performance.

Recently, various visualization techniques for malware analysis have been proposed to enable human analysts to visually observe the features of malware. Studies to visualize malware behaviors have been also conducted. Trinius et al. [13] collected information on API (Application Program Interface) calls and instructions of a certain behaviors, and they visualized the percentages of API calls into a "Treemap" as well as malware behaviors into a "Thread graph." Saxe et al. [14] proposed a system to visualize the relationships and similarities of system call sequences. The former shows map-like visualization of similarity, and the latter shows similarities and differences between selected samples, based on system calls or function calls.

Conti et al. [15] proposed an integrated visualizing system that enables the analysis of byte information of malware samples through different graphical elements. A “byteview visualization” shows each byte in the binary sample to a pixel, and a “byte presence visualization” shows how many bytes have appeared. Moreover, a “dot plot visualization” detects duplicated sequence 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. [16] visually showed the results of similarity calculations between malware samples through images named “Heatmap.” Nataraj et al. [17] scanned all malware bytes, converted the information into gray-scale images, and classified the malware using image processing. After generating images, they applied an abstract representation technique for the scene image, i.e. GIST [18,19], to compute texture features and to classify malware. Moreover, they proved that the binary texture analysis techniques using image processing can classify malware more quickly than existing malware classification methods [20]. However, since the texture analysis method has large computational overheads, the proposed method has problems to process a large number of malware.

In this paper, we propose a novel analysis method using image matrices in order to visually represent malware so that the features of malware can be easily detected and the similarities between different malware can be calculated faster than other visualization methods.

3. OUR PROPOSED METHOD

3.1 Overview

Our proposed visualized malware analysis method consists of three steps, as shown in Figure 1. In Step 1, binary information is extracted from binary sample files, and image matrices in which the binary information is recorded as RGB colored pixels are generated in Step 2. In Step 3, the similarities between the image matrices are calculated. In the following sections, each step was explained in detail.

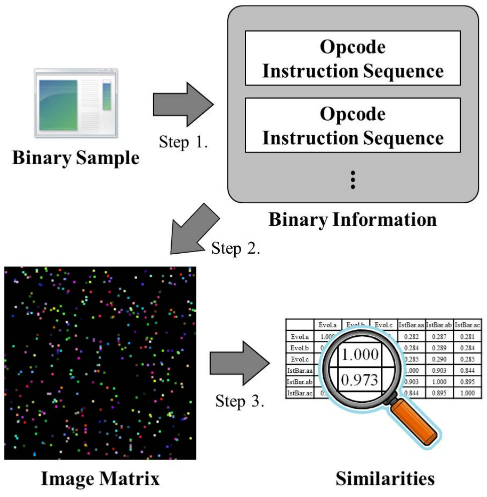


Figure 1. Overview of the proposed method

3.2 Extraction of Binary Information

Figure 2 shows the process to extract binary information from binary sample files in Step 1.

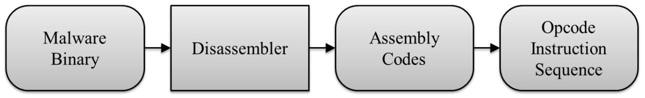
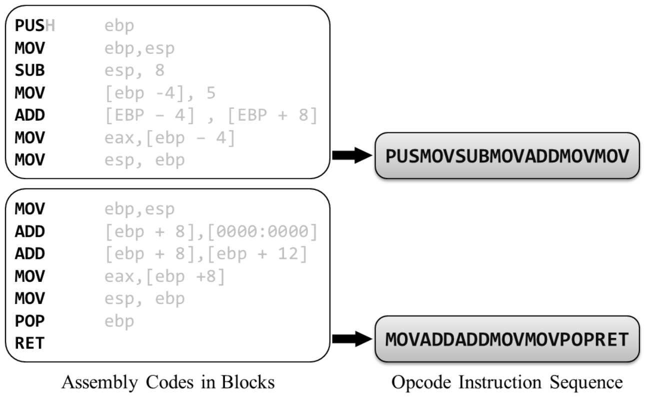


Figure 2. Binary information extraction procedure

To extract binary information, the binary sample files are disassembled first, using disassembling tools, such as IDA Pro [21] or OllyDbg [22]. After assembly codes are extracted using a tool, the sequence of assembly codes are divided into blocks according to some instructions that are used as delimiters, as shown in Figure 3.

The sequence of opcodes included in individual blocks is used as binary information. From each opcode, only first three characters are used to generate information for the block. For example, four-character opcode instructions such as `push` are reduced to three-character instruction. Then, these three-character instructions are concatenated together, and the character string is used to represent the opcode block in the next step to generate an image matrix.



```
PUSH    ebp
MOV     ebp,esp
SUB     esp, 8
MOV     [ebp -4], 5
ADD     [EBP - 4] , [EBP + 8]
MOV     eax,[ebp - 4]
MOV     esp, ebp
```

```
PUSMOVSUBMOVADDMOVMOV
```

```
MOV     ebp,esp
ADD     [ebp + 8],[0000:0000]
ADD     [ebp + 8],[ebp + 12]
MOV     eax,[ebp +8]
MOV     esp, ebp
POP     ebp
RET
```

```
MOVADDADDMOVMOVPOPRET
```

Figure 3. Opcode instructions used as binary information

3.3 Generation of Image Matrices

Figure 4 shows a procedure of Step 2 that converts the opcode instruction sequences into an image matrix. Two hash functions are used to decide coordinate information and RGB color information, as shown in Figure 4.

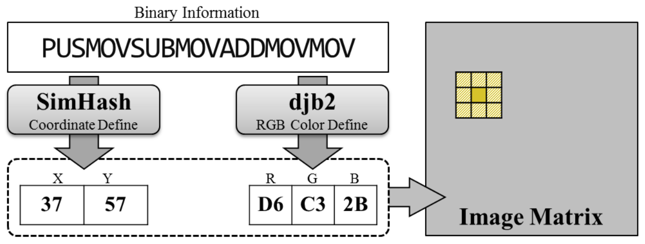


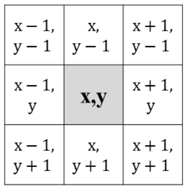
Figure 4. Generating images using binary information

In order to visualize a binary file as an image matrix, both the length and the width of an image matrix are initialized to 2^n, where n is selected by users. To reduce the probability of collisions of hash functions, n should be large enough. In our experiments, we selected n as 8 to avoid collisions.

The coordinate-defining module and the RGB color-defining module are used to generate image matrices. First, the coordinate-defining module defines the (x,y) coordinates on image matrices for binary information of each code block. SimHash [23] is applied to binary information extracted in the Step 1. SimHash is a local-sensitive hash function that assumes if input values are similar, output values will also be similar. Therefore, if character strings of binary information are similar, the outputs will be similar and it will map into similar coordinates in an image matrix.

Second, the RGB color-defining module defines the color values of images on an image matrix. djb2 [24] is applied to binary information to determine colors of images for the binary information. RGB colors are defined by calculating values of 8 bits each for red, green, and blue colors.

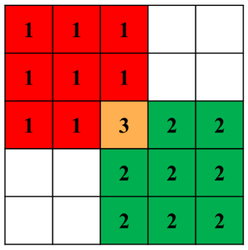
Once the coordinates and RGB colors of individual images have been defined, RGB colored images are recorded on individual coordinates of image matrices. To provide human analysts with a more convenient visual analysis, pixels around the defined coordinates are recorded simultaneously. As shown in Figure 5, nine pixels from (x–1,y–1) to (x+1,y+1) around an (x,y) coordinate defined through the opcode instruction sequence for a block are recorded.



x − 1, y − 1	x, y − 1	x + 1, y − 1
x − 1, y	**x,y**	x + 1, y
x − 1, y + 1	x, y + 1	x + 1, y + 1

Figure 5. Nine pixels recording for one opcode instruction sequence

If images are overlapped each other because coordinates defined for multiple opcode instruction sequences are adjacent, as shown in Figure 6, the sums of RGB colors become new pixel colors. If a result of color summing exceeds 255(0xFF), the result will be set to 255. For example, if RGB_1 is (255,0,0) and RGB_2 is (0,176,50), new color will become (255,176,50).



R_3 = min(R_1+R_2, FF) = min(FF+00, FF) = FF
G_3 = min(G_1+G_2, FF) = min(00+B0, FF) = B0
B_3 = min(B_1+B_2, FF) = min(00+50, FF) = 50
∴ **RGB_3 = (FF,B0,50)**

Figure 6. Method of recording overlapping pixels

Since the number of pixels recorded on an image matrix varies according to the file sizes and the number of opcode instruction sequences, and the number of overlapping pixels will increases as the number of images increases. If there are too many overlapped images, the size of the image matrix should be increased.

3.4 Similarity Calculation between Image Matrices

We used "selective area matching" to calculate the similarities between image matrices. For selective area matching, an image matrix should be divided into N pieces, where N can be set to 4^x (x=1,2,3, …), such as 4, 16, and 64. Figure 7 shows image matrices in which the areas were divided according to different N values. Then, n pieces are randomly selected from the image matrix to be compared. For example, as shown in Figure 8, an image matrix can be divided into 16 (N=16) areas and four (n=4) areas can be randomly selected.

Matching pixels are now identified in each selected area and used in similarity calculations. In this paper, vector angular-based distance measure algorithm [25] which decides the similarity using vector value for each pixel is used to calculate the similarities between image matrices. The similarities among n pieces of areas are calculated, and the overall similarity is calculated as the average of the similarities for the matching pixels on each area.

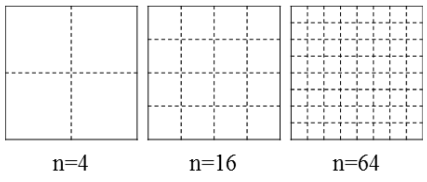


Figure 7. The areas of image matrices divided according to the values of N

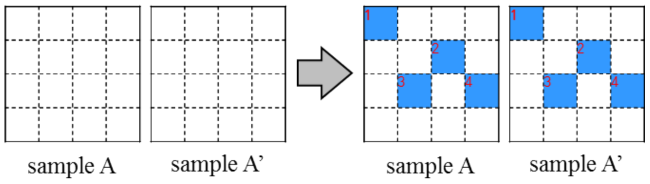


Figure 8. Examples of randomly selected areas (N=16, n=4)

4. EXPERIMENTAL RESULTS

4.1 Experimental Data

Using the visualization analysis tools implemented in this paper, image matrices were generated for the benign and malware binary samples shown in Table 1. We set the sizes of the generated image matrices to 256 × 256 pixels.

Table 1. Benign and malware binary samples

Type	File Name		# of Blocks
Benign	notepad		1072
	winmine		360
	wuauclt		1073
Malware	Boxed	.a	1038
		.b	1043
		.d	1050
	Klez	.f	1474
		.g	1475
		.j	1529
	Evol	.a	184
		.b	187
		.c	189

4.2 Results of Image Matrix Extraction

Figure 9 shows image matrices extracted from individual benign binary samples, and Figure 10 shows image matrices extracted from individual malware families. Since the number of opcode instruction sequencesused as binary information varies, the number of pixels recorded on image matrices differs. For the benign binaries, even if pixels are recorded on the same coordinates of different image matrices, similarities between image matrices are minimal because the RGB color information of the relevant pixels is different. In contrast, many similar pixels are found among the image matrices of binary files classified to the same malware family.

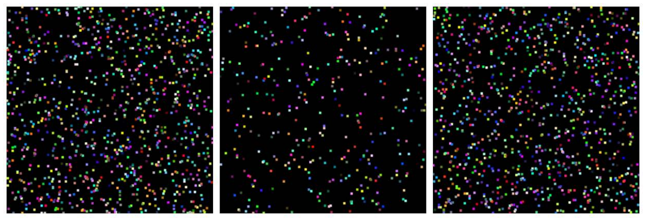


Figure 9. Image matrices of benign binaries

Figure 10 shows the image matrices of Trojan-DDoS.Win32. Boxed, Email-Worm.Win32.Klez and Virus.Win32.Evol families, respectively. Same RGB colored pixels recorded in the same coordinate can be found in the image matrices extracted from variants of the same malware families.

Images for 1038, 1043, and 1050 opcode instruction sequences were respectively recorded on the malware Boxed family's image matrices, as shown in Figure 10(a), and 731 images were the same. For the malware Klez family in Figure 10(b), images for 1474, 1475, and 1529 opcode instruction sequences were respectively recorded, and 984 images were the same. For the malware Evol family in Figure 10(c), images for 184, 187, and 189 opcode instruction sequences were respectively recorded, and 102 images were the same.

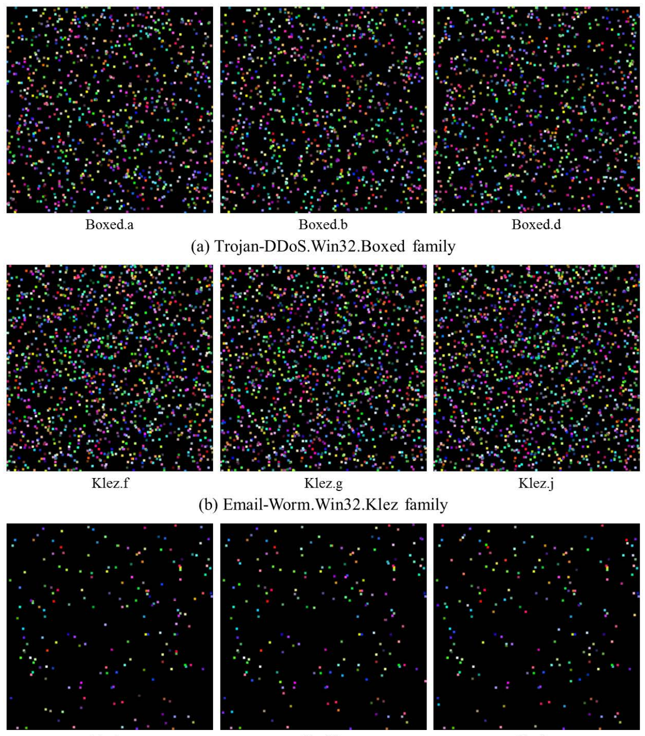

Boxed.a
Boxed.b
Boxed.d
(a) Trojan-DDoS.Win32.Boxed family
Klez.f
Klez.g
Klez.j
(b) Email-Worm.Win32.Klez family
Evol.a
Evol.b
Evol.c
(c) Virus.Win32.Evol family

Figure 10. Image matrices of the malware families

In contrast, among several hundred opcode instruction sequences on different image matrices extracted from benign binaries, only 9 cases showed the same RGB colored images from the same coordinates and a maximum of 18 cases showed the same images on image matrices of benign binaries and malware families. Our results show that image matrices of variants included in same malware family can be shown to be similar and that clear differences exist between malware binaries and benign binaries.

4.3 Results of Image Matrix Similarity

Table 2 shows the results of similarity calculations between image matrices extracted from binary samples of three malware families. The image matrices from the same malware family have at least 0.95 similarity on average, but, those from different families have 0.325 similarity on average. Figure 11 shows the results of average similarities among fifty malware sample files from ten malware families. The average similarity within the same family is 0.984; whereas, the average similarity between different families is 0.309. Therefore, our proposed method can be used to classify malware family effectively. The average time spent to calculate the similarities between image matrices was about 2.4 ms.

Table 2. Similarities between image matrices of malware

		Boxed family			Klez family			Evol family		
		.a	.b	.d	.f	.g	.j	.a	.b	.c
Boxed family	.a	1	0.968	0.936	0.450	0.449	0.443	0.266	0.267	0.267
	.b	0.968	1	0.932	0.449	0.449	0.442	0.266	0.266	0.267
	.d	0.936	0.932	1	0.448	0.448	0.442	0.265	0.266	0.266
Klez family	.f	0.450	0.449	0.448	1	0.999	0.965	0.263	0.263	0.266
	.g	0.449	0.449	0.448	0.999	1	0.965	0.263	0.263	0.264
	.j	0.443	0.442	0.442	0.965	0.965	1	0.263	0.263	0.264
Evol family	.a	0.266	0.266	0.265	0.263	0.263	0.263	1	0.986	0.960
	.b	0.267	0.266	0.266	0.263	0.263	0.263	0.986	1	0.974
	.c	0.267	0.267	0.266	0.264	0.264	0.264	0.960	0.974	1

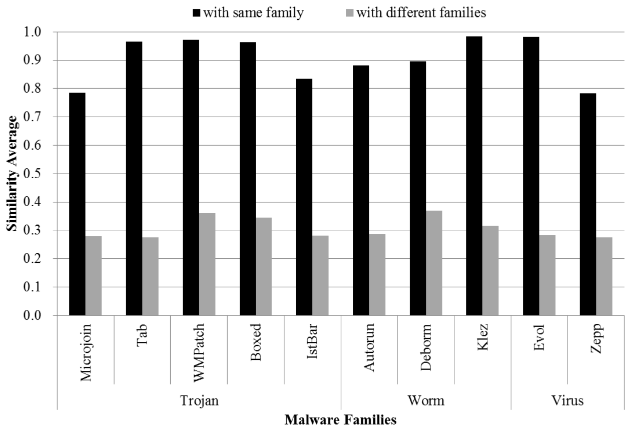


Figure 11. Average similarities of image matrices from the 10 malware families

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel method to visually analyze malware by generating image matrices from assembly instructions. The proposed method was implemented with visualization analysis tools. The experimental results showed that binary variants included in the same malware family were similar when converted into image matrices. The similarities were calculated through selective area matching, and the similarities of malware variants were shown to be higher. With our proposed method, malware analysts can analyze malware files visually, and can distinguish similar malware files for further analysis.

Our future studies include visualizaing various other information from binary files, and extending opcode instruction sequences and algorithms for automatic malware classification.

6. ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Plannig (2011-0029924)

7. REFERENCES

[1] Christodorescu, M. and Jha, S., 2004. Testing malware detectors. *ACM SIGSOFT Software Engineering Notes 29*, 4, 34-44.

[2] Kang, B., Kim, T., Kwon, H., Choi, Y., and Im, E.G., 2012. Malware classification method via binary content comparison. In *Proceedings of the 2012 ACM Research in Applied Computation Symposium* ACM, 316-321.

[3] Moser, A., Kruegel, C., and Kirda, E., 2007. Limits of static analysis for malware detection. In *Proceedings of the Twenty-Third Annual IEEE Computer Security Applications Conference (ACSAC) 2007.*, 421-430.

[4] Cesare, S. and Xiang, Y., 2010. A fast flowgraph based classification system for packed and polymorphic malware on the endhost. In *Proceedings of the 24th IEEE International Conference on IEEE Advanced Information Networking and Applications (AINA), 2010*, 721-728.

[5] Kinable, J. and Kostakis, O., 2011. Malware classification based on call graph clustering. *Journal in computer virology 7*, 4, 233-245.

[6] Shang, S., Zheng, N., Xu, J., Xu, M., and Zhang, H., 2010. Detecting malware variants via function-call graph similarity. In *Proceedings of the 5th International Conference on IEEE Malicious and Unwanted Software (MALWARE), 2010*, 113-120.

[7] Tabish, S.M., Shafiq, M.Z., and Farooq, M., 2009. Malware detection using statistical analysis of byte-level file content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, ACM, 23-31.

[8] Bilar, D., 2007. Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics* 1, 2, 156-168.

[9] Han, K.S., Kim, S.-R., and Im, E.G., 2012. Instruction frequency-based malware classification method. *INFORMATION - An International Interdisciplinary Journal 15*, 7, 2973-2984.

[10] Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C., and Bringas, P.G., 2010. Idea: Opcode-sequence-based malware detection. *Engineering Secure Software and Systems*, Springer, 35-43.

[11] Sung, A.H., Xu, J., Chavez, P., and Mukkamala, S., 2004. Static analyzer of vicious executables (save). In *Proceedings of the 20th Annual IEEE Computer Security Applications Conference, 2004.*, 326-334.

[12] Walenstein, A., Venable, M., Hayes, M., Thompson, C., and Lakhotia, A., 2007. Exploiting similarity between variants to defeat malware. In *Proceedings of the BlackHat DC Conference*.

[13] Trinius, P., Holz, T., Gobel, J., and Freiling, F.C., 2009. Visual analysis of malware behavior using treemaps and thread graphs. In *Proceedings of the 6th International Workshop on IEEE Visualization for Cyber Security (VizSec) 2009.*, 33-38.

[14] Saxe, J., Mentis, D., and Greamo, C., 2012. Visualization of shared system call sequence relationships in large malware corpora. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security*, ACM, 33-40.

[15] Conti, G., Dean, E., Sinda, M., and Sangster, B., 2008. Visual reverse engineering of binary and data files. *Visualization for Computer Security*, Springer, 1-17.

[16] Anderson, B., Storlie, C., and Lane, T., 2012. Improving malware classification: bridging the static/dynamic gap. In *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, ACM, 3-14.

[17] Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B., 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security,* ,ACM.

[18] Oliva, A. and Torralba, A., 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42, 3, 145-175.

[19] Torralba, A., Murphy, K.P., Freeman, W.T., and Rubin, M.A., 2003. Context-based vision system for place and object recognition. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 273-280.

[20] Nataraj, L., Yegneswaran, V., Porras, P., and Zhang, J., 2011. 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*, ACM, 21-30.

[21] Eagle, C., 2008. The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler. No Starch Press.

[22] Yuschuk, O., 2007. Ollydbg. http://www.ollydbg.de/

[23] Charikar, M.S., 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, ACM, 380-388.

[24] D. Bernstein. Usenet posting, comp.lang.c. http://groups.google.com/group/comp.lang.c/msg/6b82e964887d73d9, Dec. 1990.

[25] Androutsos, D., Plataniotis, K., and Venetsanopoulos, A.N., 1999. A novel vector-based approach to color image retrieval using a vector angular-based distance measure. *Computer Vision and Image Understanding*,75, 1, 46-58.