Malware Images: Visualization and Automatic Classification

Article · July 2011

DOI: 10.1145/2016904.2016908

4 authors:

Lakshmanan Nataraj
Mayachitra Inc.
31 PUBLICATIONS **722** CITATIONS
SEE PROFILE

Shanmugavadivel Karthikeyan
Synaptics
25 PUBLICATIONS **484** CITATIONS
SEE PROFILE

Grégoire Jacob
20 PUBLICATIONS **845** CITATIONS
SEE PROFILE

B. S. Manjunath
University of California, Santa Barbara
449 PUBLICATIONS **24,412** CITATIONS
SEE PROFILE

Malware Images: Visualization and Automatic Classification

L. Nataraj, S. Karthikeyan,
Dept. of Electrical and Computer Engineering,
University of California, Santa Barbara
lakshmanan_nataraj,karthikeyan@ece.ucsb.edu

G. Jacob,
Dept. of Computer Science,
University of California, Santa Barbara
gregoire.jacob@gmail.com

B. S. Manjunath
Dept. of Electrical and Computer Engineering,
University of California, Santa Barbara
manj@ece.ucsb.edu

ABSTRACT
We propose a simple yet effective method for visualizing and classifying malware using image processing techniques. Malware binaries are visualized as gray-scale images, with the observation that for many malware families, the images belonging to the same family appear very similar in layout and texture. Motivated by this visual similarity, a classification method using standard image features is proposed. Neither disassembly nor code execution is required for classification. Preliminary experimental results are quite promising with 98% classification accuracy on a malware database of 9,458 samples with 25 different malware families. Our technique also exhibits interesting resilience to popular obfuscation techniques such as section encryption.

Categories and Subject Descriptors
D.4.6 [**Security and Protection**]: *Invasive Software (viruses, worms, Trojan horses)*

I.4 [**Image Processing and Computer Vision]:** *Applications*

I.5 [**Pattern Recognition]**: *Applications*

H.1.2 [**User/Machine Systems]:** *Human Information Processing*

General Terms
Computer Security, Visualization, Malware, Image Processing,

Keywords
Malware Visualization, Image Texture, Malware Classification

1. INTRODUCTION
Traditional approaches towards analyzing malware involve extraction of binary signatures from malware, constituting their fingerprint. Due to the rapid proliferation of malware, there is an exponential increase in the number of new signatures released every year (in [1], Symantec reported 2,895,802 new signatures in 2009, as compared to 169,323, in 2008).

Other approaches of analyzing malware include static code analysis and dynamic code analysis. Static analysis works by disassembling the code and exploring the control flow of the executable to look for malicious patterns. On the other hand, dynamic analysis works by executing the code in a virtual environment and a behavioral report characterizing the executable is generated based on the execution trace. Both these techniques have their pros and cons. Static analysis offers the most complete coverage but it usually suffers from code obfuscation. The executable has to be unpacked and decrypted before analysis, and even then, the analysis can be hindered by problems of intractable complexity. Dynamic analysis is more efficient and does not need the executable to be unpacked or decrypted. However, it is time intensive and resource consuming, thus raising scalability issues. Moreover, some malicious behaviors might be unobserved because the environment does not satisfy the triggering conditions.

In this paper, we take a completely different and novel approach to characterize and analyze malware. At a broader level, a malware executable can be represented as a binary string of zeros and ones. This vector can be reshaped into a matrix and viewed as an image. We observed significant visual similarities in image texture for malware belonging to the same family. This perhaps could be explained by the common practice of reusing the code to create new malware variants. In Sec.3 we discuss representing malware binaries as images. In Sec.4 we consider malware classification problem as one of image classification. Existing classification techniques require either disassembly or execution whereas our method does not require either but still shows significant improvement in terms of performance. Further, it is also resilient to popular obfuscation techniques such as section encryption. This automatic classification technique should be very valuable for anti-virus companies and security researchers who receive hundreds of malware everyday.

The rest of this paper is organized as follows. In Sec. 2, we discuss the related work in malware visualization and classification. In Sec.3 and Sec.4, we describe our method to visualize malware and automatically classify them using images. The experiments are detailed in Sec. 5. We discuss the limitations of our approach in Sec. 6 and conclude in Sec.7.

VizSec'11, July 20, 2011, Pittsburg, PA, USA.

2. RELATED WORK

Several tools such as text editors and binary editors can both visualize and manipulate binary data. Of late, there have been several GUI-based tools which facilitate comparison of files. However, there has been limited research in visualizing malware. In [3] Yoo used self organizing maps to detect and visualize malicious code inside an executable. In [4] Quist and Liebrock develop a visualization framework for reverse engineering. They identify functional areas and de-obfuscate through a node-link visualization where nodes represent the address and links represent state transitions between addresses. In [5] Trinius et al. display the distributions of operations using treemaps and the sequence of operations using thread graphs. In [6] Goodall et al. develop a visual analysis environment that can aid software developers to understand the code better. They also show how vulnerabilities within software can be visualized in their environment.

While there hasn't been much work on viewing malware as digital images, Conti et al. [8,9] visualized raw binary data of primitive binary fragments such as text, C++ data structure, image data, audio data as images. In [7] Conti et al. show that they can automatically classify the different binary fragments using statistical features. However, their analysis is only concerned with identifying primitive binary fragments and not malware. This work presents a similar approach by representing malware as grayscale images.

Several techniques have been proposed for clustering and classification of malware. These include both static analysis [13-19] as well as dynamic analysis [20-24]. We will review papers that specifically deal with classification of malware. In [24] Rieck et al. used features based on behavioral analysis of malware to classify them according to their families. They used a labeled dataset of 10,072 malware samples labeled by an anti-virus software and divide the dataset into 14 malware families. Then they monitored the behavior of all the malware in a sandbox environment which generated a behavioral report. From the report, they generate a feature vector for every malware based on the frequency of some specific strings in the report. A Support Vector Machine is used for training and testing the feature on the 14 families and they report an average classification accuracy of 88%. In contrast to [24], Tian et al [16] use a very simple feature, the length of a program, to classify 7 different types of Trojans and obtain an average accuracy of 88%. However, their analysis was only done on 721 files. In [17,18] the same authors improve their above technique by using printable string information from the malware. They evaluated their method on 1521 malware consisting of 13 families and reported a classification accuracy of 98.8%. In [20], Park et al. classify malware based on detecting the maximal common sub graph in a behavioral graph. They demonstrate their results on a set of 300 malware in 6 families.

With respect to related works, our classification method does not require any disassembly or execution of the actual malware code. Moreover, the image textures used for classification provide more resilient features in terms of obfuscation techniques, and in particular for encryption. Finally, we evaluated our approach on a larger dataset consisting in 25 families within a malware corpus of 9,458 malware. The evaluation results show that our method offers similar precision at a lower computational cost.

3. VISUALIZATION

A given malware binary is read as a vector of 8 bit unsigned integers and then organized into a 2D array. This can be visualized as a gray scale image in the range [0,255] (0: black, 255: white). The width of the image is fixed and the height is allowed to vary depending on the file size (Fig. 1). Tab. 1 gives some recommended image widths for different file sizes based on empirical observations.

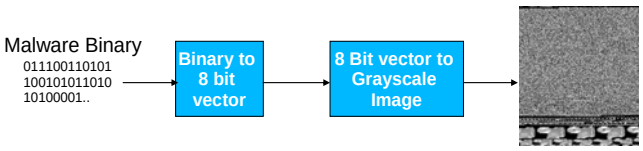

Fig.1 Visualizing Malware as an Image

Fig. 2 shows an example image of a common Trojan downloader, Dontovo A, which downlods and executes arbitrary files [26]. It is interesting to note that in many cases, as in Fig. 2, different sections (binary fragments) of the malware exhibit distinctive image textures. A detailed taxonomy of various primitive binary fragments and their visualization as grayscale images can be found in [9].

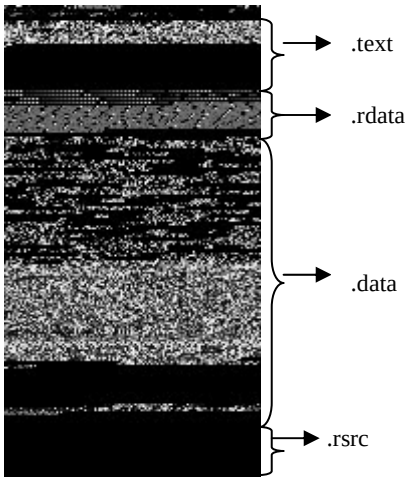

Fig. 2 Various Sections of Trojan: Dontovo.A

The .text section contains the executable code. From the figure, we can see that the first part of the .text section contains the code whose texture is fine grained. The rest is filled with zeros (black) indicating zero padding at the end of this section. The following .data section contains both uninitialized code (black patch) and initialized data (fine grained texture). The final section is the .rsrc section which contains all the resources of the module. These may also include icons that an application may use.

Tab. 1: Image Width for Various File Sizes

File Size Range	Image Width
<10 kB	32
10 kB – 30 kB	64
30 kB – 60 kB	128
60 kB – 100 kB	256
100 kB – 200 kB	384
200 kB – 500 kB	512
500 kB – 1000 kB	768
>1000 kB	1024

4. MALWARE CLASSIFICATION

Fig. 3 shows examples of malware from two different families. An empirical observation one can make here is that images of different malware samples from a given family appear visually similar and distinct from those belonging to a different family. As noted earlier, this can perhaps be attributed to re-use of old malware binaries to create new ones. The visual similarity of malware images motivated us to look at malware classification using techniques from computer vision, where image based classification has been well studied. The images of specific families of malware can be seen in Fig. 7. As can be seen from Fig.7, various malware families have distinct visual characteristics.

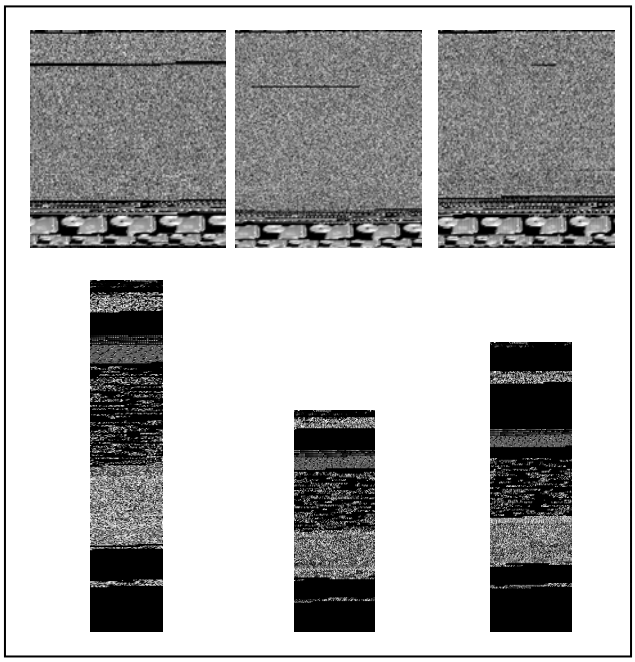

Fig. 3 The images in the first row are images of 3 instances of malware belonging to the family Fakerean [26] and those in the second row belong to the family Dontovo.A [26].

4.1 Image Texture

There is no commonly accepted definition of what visual texture means, but it often is associated with (repeated) patterns such as those shown in Fig 4 [27]. Three of the main areas on texture research are texture classification, texture analysis and texture synthesis. Texture classification is concerned identifying various uniformly textured regions in images. Identifying the boundaries of various texture regions is the main goal of texture segmentation. Texture synthesis methods are used to synthesize texture images. They are frequently used in computer graphics.

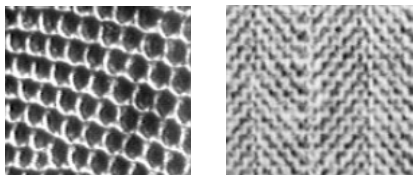

Fig. 4 Examples of two texture images from Brodatz's album [28]

Texture analysis is an important area of study in computer vision. Most surfaces exhibit some amount of texture. Texture analysis is used in many applications including medical image analysis, remote sensing, and document image processing. The malware pictures shown earlier in Fig 2-3, though not exactly are repeated patterns, exhibit significant amount of "texture" and this information can be exploited for automated classification.

4.2 Feature Vector and Classifier

Several features have been proposed to analyze texture. One of the most common methods of texture analysis is analyzing the frequency content of a texture block. Standard approaches divide the frequency domain into rings (scale) and wedges (orientations) and features are computed in these regions. Psychophysical results have shown that the human eye analyzes texture by decomposing the image into its frequency and orientation components. A popular computational approach to texture analysis is using Gabor filtering. A two dimensional Gabor function consists of a sinusoidal plane of certain frequency and orientation that is modulated by a Gaussian envelope. A Gabor filter is a filter that is frequency and orientation selective. By varying the frequencies and orientations, we obtain a bank of Gabor filters. An image is passed through this bank of filters to obtain several filtered images from which texture based features are extracted. One such feature is obtained by computing the absolute average deviation of the transformed values from the filtered images from a mean within a small window. Texture features using Gabor filters have been successful in texture segmentation and classification.

We use a similar feature in this paper to characterize and classify malware. To compute texture features, we use GIST [11],[12] which uses a wavelet decomposition of an image. This feature has been successful in scene classification and object classification. Each image location is represented by the output of filters tuned to different orientations and scales. We use a steerable pyramid with 8 orientations and 4 scales applied to the image. The local representation of an image is then given by: $v^L(x) = \{v_k(x)\}_{k=1,N}$ where *N=20* is the number of sub-bands. In order to capture global image properties while retaining some

local information, we compute the mean value of the magnitude of the local features averaged over large spatial regions:

$$m(x) = \sum_{x'} |v(x')| w(x' - x) \quad (1)$$

where $w(x)$ is the averaging window. The resulting representation is downsampled to have a spatial resolution of *MxM* pixels (here we use *M*=4). Thus, m has size *M x M x N* = 320 which is the dimension of the GIST feature we use. A more detailed explanation on GIST features can be found in [12].

We use k-nearest neighbors with Euclidean distance for classification. For all our tests, we do a 10 fold cross validation, where under each test, a random subset of a class is used for training and testing. For each iteration, this test randomly selects 90% data from a class for training and 10% for testing. Hence, a given test data is classified to the class which is the mode of its k nearest neighbors.

5. EXPERIMENTS

In this section, the malware we examined are malware executables submitted to the Anubis analysis system [2]. The tested samples are thus recent malware that can be found "in the wild". To obtain the ground truth for our tests, we classify them into different malware families using the labels provided by Microsoft Security Essentials.

5.1 Hypothesis Validation

In order to validate the hypothesis that malware families exhibit some visual similarities, we first picked a smaller dataset consisting of 8 malware families, totaling 1713 malware images. We went through the thumbnails of these images and verified that the images belonging to a family were indeed similar. GIST image features are computed for each of these images. The average time to compute the Gist feature on an image is 54 ms. The high-dimensional GIST features are then projected to a lower dimensional space for visualization/analysis [10]. As shown in Fig. 5, the feature points for families Allaple.A, VB.AT, Wintrim.BX, Yuner.A and Fakerean are well separated. However, there seems to be confusion amongst families Instantaccess, Obfuscator.AD and Skintrim.N. This is also evident from their grayscale visualizations shown in Fig.7 and they appear very similar to the human eye as well. However, these families are still classified accurately with our classification method. We then use a k-nearest neighbor (k=3) classifier using 10 fold cross validation for classification and obtain an classification rate of 0.9993, averaged over 10 tests with a standard deviation of 0.0019. The confusion matrix is shown in Tab.2. Varying k between 1 and 10 gave similar results although k=3 gave the best accuracy. These tests are repeated after adding to the set an additional 123 benign executables from the Win32 system files and applications. The dataset we used can be obtained from [30]. With the new dataset, the classification rate was 0.9929 over a 10 fold cross validation with a standard deviation of 0.002.

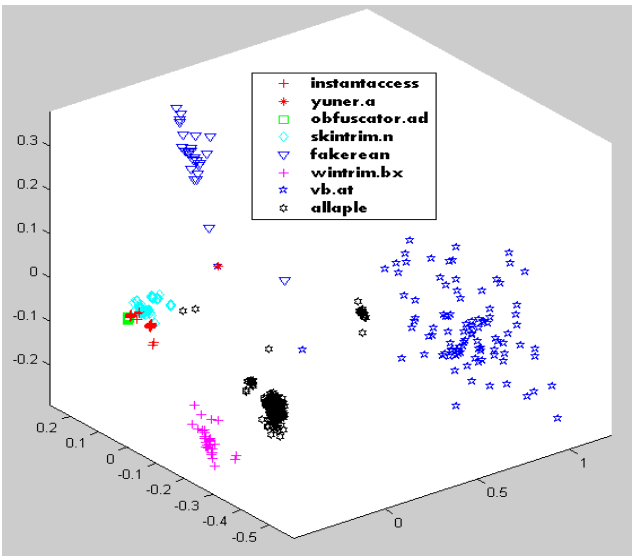

Fig.5 GIST Features projected in lower dimensions using multidimensional scaling [10]

Tab2. Confusion Matrix for classification using GIST Features

	A	B	C	D	E	F	G	H
A	**1**	0	0	0	0	0	0	0
B	0	**1**	0	0	0	0	.01	0
C	0	0	**1**	0	0	0	0	0
D	0	0	0	**1**	0	0	0	0
E	0	0	0	0	**.997**	0	.01	0
F	0	0	0	0	0	**1**	0	0
G	0	0	0	0	0	0	**.98**	
H	0	0	0	0	.003	0	0	**1**

The malware families in this experiment include 335 of Instantaccess (A), 485 of Yuner.A (B), 111 of Obfuscator.AD (C), 80 of Skintrim.N (D), 298 of Fakerean (E), 88 of Wintrim.BX (F), 97 of VB.AT (G) and 219 of Allaple.A (H).

5.2 Large Scale Experiments

We now extend our analysis to a larger dataset consisting of 25 malware families, totaling 9,458 malware, see Tab.3 for more details. Malware belonging to families Yuner.A, VB.AT, Malex.gen!J, Autorun.K, Rbot!gen, were packed (UPX). These are unpacked for preliminary analysis. The above tests are then repeated to obtain a classification accuracy of 0.9718 for the 25 malware families. The images of these families can be obtained from [30]. The confusion matrix is shown in Fig. 6(a). As seen in Fig.6(a), there is confusion between the families such as C2Lop.P, C2Lop.gen!g and Swizzor.gen!I, Swizzor.gen!E. These are variants of C2Lop and Swizzor respectively. If these families are combined together as one, the recomputed accuracy is 0.992 and the corresponding confusion matrix is shown in Fig. 6(b). On adding an extra set of benign executables, the accuracy still remained high at 0.9808.

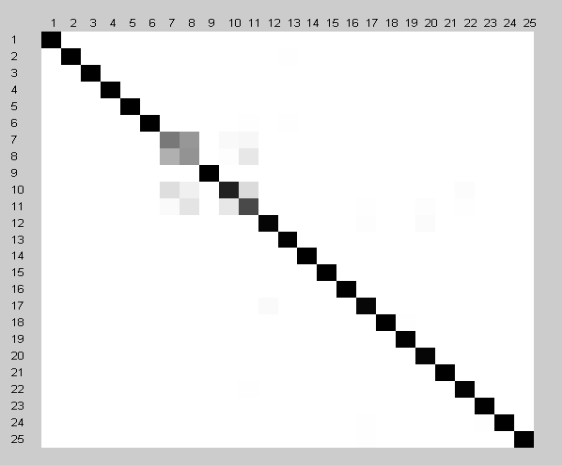

Fig.6 (a) Confusion matrix with confusion among variants.

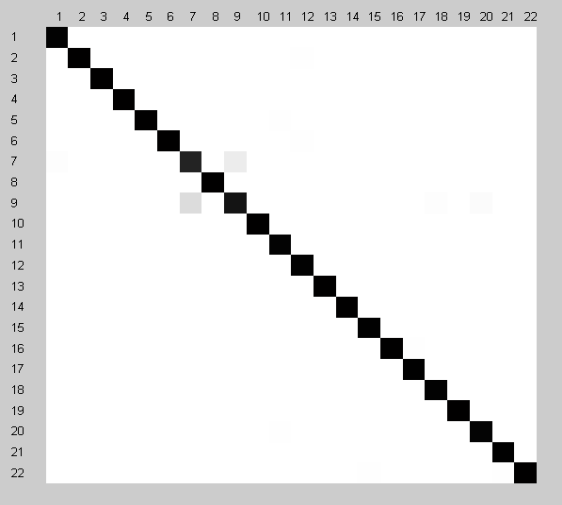

Fig. 6 (b) Variants combined as one family.

5.3 Resilience to Obfuscation

The analysis so far has been on unpacked executables. Packing alters the structure of the binary and hence the new binary after packing no longer appears like the unpacked one. However, when malware belonging to the same family are packed with the same packer, we conjecture that the images of the packed malware appear similar. Hence, in this experiment, we treat the packed malware of a family as a new family and repeat the above experiment. The total number of families is now 29. Our technique is still able to classify the different families even after packing and the overall classification accuracy is 0.9808.

To avoid the use of packers, certain malware families directly embed a polymorphic engine within their code. The most representative example in our experiments is Allaple where the code section is encrypted in several layers, using random keys [29]. Nonetheless, the proposed method is able to classify these samples as well. Our approach can successfully handle section encryption because it mainly relies on textural information which is preserved by the weak encryption schemes used by polymorphic engines.

5.4 Performance Comparison

Looking at related works on classification using static features, classification based on bi-gram extraction seems the prevalent method, such as in [13]. To measure the performance gain brought by our approach, we extract the bi-grams distributions from our first dataset of 8 families. Bi-grams are computed directly from the raw data without any disassembly, which would have been even slower. Using these distributions as feature vectors, we obtained a classification accuracy of 0.98, which is similar to our approach. However, the average extraction time is 5s and the time taken to classify a sample is 56s. In contrast, the time taken to compute GIST feature is 54ms and the overall classification time was 1.4s. The proposed method is about 40 times faster, and is partly explained by the fact that the feature vector length used to characterize a malware image is about 320 whereas about 65K elements are needed for the distribution based analysis using the bi-grams.

6. LIMITATIONS AND FUTURE WORK

Although an image processing based approach is a novel approach to analyze malware, an adversary who knows the technique can take countermeasures to beat the system since our technique is based on global image based features. Some examples of countermeasures could be relocating sections in a binary or adding vast amount of redundant data. To tackle against such attacks, we will explore more localized feature extraction schemes that take into account the distinct characteristics of malware executables and their primitive binary segments [8, 9]. One possible future extension is to segment out the image regions, and characterize the local texture and spatial distribution of these texture patterns.

Another area of future work is clustering of malware using image based features. Although clustering and classification are similar, the former is unsupervised and the latter is supervised. Current state of the art clustering algorithms [21],[22] report highly accurate results in terms of precision and recall. However, in a recent paper, Li et al.[25] did a performance evaluation of the current clustering algorithms and concluded that high accuracy is due to the selection bias in the ground truth data. Hence, the results reported in state of the art clustering papers are not reliable.

7. CONCLUSIONS

This paper presented a novel approach to malware analysis based on visualizing and processing malware as images. A commonly used image feature descriptor is used to characterize the malware globally. The preliminary results are very encouraging, with high accuracy classification that is competitive with the state of the art results in the literature at a significantly less computational cost. We believe that using computer vision techniques for malware analysis opens the path to a broader spectrum of novel ways to analyze malware.

8. ACKNOWLEDGEMENTS

We are grateful to authors of Anubis [2] for providing us with the malware dataset. We would also like to thank Prof. Giovanni Vigna,

Prof. Christopher Kruegel and the anonymous reviewers for their valuable feedback. This work has been supported by the grant ONR # N00014-11-10111.

Tab.3 Malware Dataset of 25 Families

#	Class	Family	#
1.	Worm	Allaple.L	1591
2.	Worm	Allaple.A	2949
3.	Worm	Yuner.A	800
4.	PWS	Lolyda.AA 1	213
5.	PWS	Lolyda.AA 2	184
6.	PWS	Lolyda.AA 3	123
7.	Trojan	C2Lop.P	146
8.	Trojan	C2Lop.gen!g	200
9.	Dialer	Instantaccess	431
10.	TDownloader	Swizzot.gen!I	132
11.	TDownloader	Swizzor.gen!E	128
12.	Worm	VB.AT	408
13.	Rogue	Fakerean	381
14.	Trojan	Alueron.gen!J	198
15.	Trojan	Malex.gen!J	136
16.	PWS	Lolyda.AT	159
17.	Dialer	Adialer.C	125
18.	TDownloader	Wintrim.BX	97
19.	Dialer	Dialplatform.B	177
20.	TDownloader	Dontovo.A	162
21.	TDownloader	Obfuscator.AD	142
22.	Backdoor	Agent.FYI	116
23.	Worm:AutoIT	Autorun.K	106
24.	Backdoor	Rbot!gen	158
25.	Trojan	Skintrim.N	80

9. REFERENCES

[1]. Symantec Global Internet Security Threat Report, April 2010.

[2]. Anubis: Analyzing Unknown Binaries, < http://anubis.iseclab.org/>

[3]. Yoo, I. Visualizing Windows Executable Viruses Using Self-Organizing Maps., 2004 *International Workshop on Visualization for Cyber Security (VizSec).*.

[4]. Quist, D.A. and Liebrock, L.M. 2009. Visualizing compiled executables for malware analysis. *International Workshop on Visualization for Cyber Security (VizSec)*, 27-32.

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

[6]. Goodall, J,H. Randwan H. and Halseth, L. 2010. Visual analysis of code Security. In *International Workshop on Visualization for Cyber Security (VizSec).*

[7]. Conti, G. Bratus, S. Sangster, B. Ragsdale, S. Supan, M. Lichtenberg, A. Perez, R. and Shubina, A. 2010. Automated Mapping of Large Binary Objects Using Primitive Fragment Type Classification *Digital Forensics Research Conference (DFRWS)*

[8]. Conti, G. and Bratus, S. 2010. Voyage of the Reverser: A Visual Study of Binary Species, *Black Hat USA.*

[9]. Conti, G. Bratus, S. Shubina, A. Lichtenberg, A. Ragsdale, R. Perez-Alemany, R. Sangster, B. and Supan, M. 2010. A Visual Study of Binary Fragment Types *Black Hat USA.*

[10]. Multi-dimensional Scaling, Dr Toolbox http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

[11] Torralba, A. Murphy, K.P. Freeman, W.T. and Rubin, M.A. 2003, Context-based vision systems for place and object recognition, *Intl. Conf. on Computer Vision (ICCV).*

[12]. Oliva, A. and Torralba. A, 2001. Modeling the shape of a scene: a holistic representation of the spatial envelope, *International Journal of Computer Vision*, Vol. 42(3), 145-175.

[13]. Karim, M. E., Walenstein, A., Lakhotia, A. & Parida, L. 2005. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, **1** (1)**:**13-23.

[14]. Kolter, J. Z. and Maloof, M. A. 2004. Learning to detect malicious executables in the wild. *International Conference on Knowledge Discovery and Data Mining*, 470-478.

[15]. Gao, D., Reiter, M. K. & Song, D. 2008. Binhunt: Automatically finding semantic differences in binary programs. *Information and Communications Security*, **5308**:238–255,

[16] Tian, R. Batten, L.M. and Versteeg. S.C. 2008. Function length as a tool for malware classification. *3rd International Conference on Malicious and Unwanted Software (MALWARE),*.

[17] Tian, R. Batten, L. Islam, R. and Versteeg, S. 2009 An automated classification system based on the strings of trojan and virus families. *4rd International Conference on Malicious and Unwanted Software: MALWARE 2009*, pages 23–30.

[18] Islam, R., Tian R., Batten, L., Versteeg, S. 2010 Classification of Malware Based on String and Function Feature Selection. *2nd Cybercrime and Trustworthy Computing Workshop.*

[19] Gheorghescu, M. 2005 An automated virus classification system. *Virus Bulletin Conference*, 294-300.

[20]. Park, Y. Reeves, D. Mulukutla, V. Sundaravel, B. 2010. Fast malware classification by automated behavioral graph matching, *Proc. Of Sixth Annual Workshop on Cyber Security and Information Intelligent Research (CSIIRW' 10)*, 2010.

[21] Bailey, M. Oberheide, J. Andersen, J..Mao, Z.M. Jahanian, F. and Nazario, J. 2007 Automated classification and analysis of internet malware. *RAID*, 4637:178–197.

[22]. Bayer, U. Milani Comparetti, P. Hlauschek, C. Kruegel, C. and Kirda, E. 2009. Scalable, behavior-based malware clustering. *NDSS'09 Security Symposium*, 2009.

[23]. Lee, T. and Mody, J.J. 2006. Behavioral classification. *EICAR 2006*.

[24]. Rieck, K. Holz, T. Willems, C. Dussel, P. and Laskov, P. Learning and classification of malware behavior. 2008. *Fifth Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'08)*, pages 108–125.

[25]. Li. P, Liu. L, Gao. D, Reiter. M, 2010. On Challenges in evaluating malware clustering, *Proc. RAID'10*

[26]. Microsoft Malware Enclyopedia,

<*http://www.microsoft.com/security/portal/Threat/Encyclopedia/Browse.aspx*>

[27]. Tuceryan, M. and Jain, A.K. 1998. Texture Analysis, *In The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*,pp. 207-248.

[28]. Brodatz, P., Textures: A Photographic Album for Artists and Designers. New York, *Dover Publications*, 1966.

[29]. Krejdl M. Inside Win32:Allaple – Avast Blog. <http://blog.avast.com/2009/05/22/inside-win32allaple/>

[30]. Malware Images. <http://vision.ece.ucsb.edu/~lakshman/malware_images/album/>

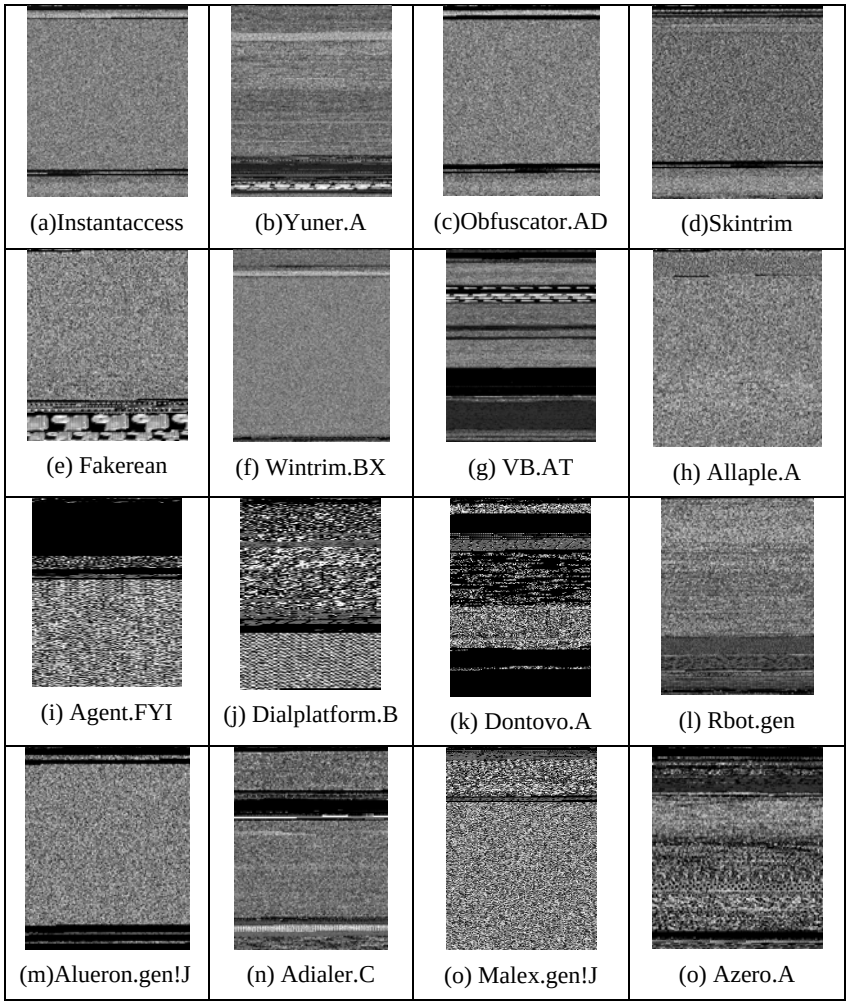

Fig. 7 Malware Images belonging to various malware families