DETECTING PACKED EXECUTABLES USING STEGANALYSIS

Colin Burgess, Fatih Kurugollu, Sakir Sezer & Keiran McLaughlin
Centre for Secure Information Technologies
Queen's University Belfast
{cburgess02; f.kurugollu; s.sezer; kieran.mclaughlin}@qub.ac.uk

ABSTRACT

This paper proposes a novel method of detecting packed executable files using steganalysis, primarily targeting the detection of obfuscated malware through packing. Considering that over 80% of malware in the wild is packed, detection accuracy and low false negative rates are important properties of malware detection methods. Experimental results outlined in this paper reveal that the proposed approach achieving an overall detection accuracy of greater than 99%, a false negative rate of 1% and a false positive rate of 0%.

1. INTRODUCTION

The world of malicious software (malware) writing has moved from harmless protest/hobbyist hackers and script kiddies to a multi-billion dollar underground industry. In order to assure their customers that their product can perform as described malware authors need to utilize obfuscation techniques to bypass detection by anti-virus vendors. The common choice in this battle is to employ executable packing. When an executable program is packed, any signatures which existed in the original binary do not exist in its packed form. A packed executable will store the contents of another program in an encrypted and compressed form within its sections. It also contains the decryption/unpacking routine which will unpack and write the original binary file into memory before executing it. Reports have shown that 80% of malware discovered by anti-virus vendors is found in packed form with 35% of those instances being packed with custom built packers [**1**].

Many studies have been carried out to classify packed and non-packed executables. The use of entropy scoring is prevalent due to its ability to detect areas of randomness in a file which are often attributable to encrypted or packed data [**2**][**3**]. Others rely on the structural differences such as section read/write/execute permissions, number of DLL imports, non-standard or flagged section names as well as various entropy based scores [**4-9**] to achieve packed file classification. However, malware authors are becoming wise to the advances in detection techniques and are employing their own tactics to circumvent discovery. Revealing a packed executable through entropy analysis is being thwarted in the Zeus family of malware by inserting repeating bytes of data in order to lower entropy scores to that of a non-packed file [**10**]. The work in [**11**] highlights how the very successful heuristic based detection methods previously mentioned can be defeated by malware authors through some simple techniques.

In order to take the upper hand in the never ending conflict with malware there is a need to develop novel detection methodologies. To accomplish this we propose the use of Steganalysis to detect packed files. Steganalysis is the study of detecting hidden communications within a digital payload be that an image, audio or video file. Within its field there are two main methods for analysis, targeted and blind. The former is specifically designed to detect a known steganographic method of hiding data and the latter is a generic method which has no a priori knowledge of the underlying data hiding techniques.

The action of packing a portable executable has similarities to the steganographic method. If both are viewed in a coarse grain fashion they receive an input dataset, an image or executable file and produce a structurally similar output with hidden embedded information. In the case of the image it is often the case that the least significant bit of a pixel value will be changed to store a hidden message. With packing, an input file is encrypted and compressed and stored within the code and data sections of the output executable. In both cases the underlying statistical data of an un-altered file differs from that of one containing hidden information. In this paper we propose converting executable files into grayscale bitmap images coupled with the use of a blind steganalysis technique [**12**] to generate features which are used in conjunction with machine learning algorithms in order to detect packed executables.

The remainder of this paper is outlined as follows; Section 2 outlines related work in the area of image processing for executable classification. The proposed methodology and experiments are described in sections 3 and 4 with results in section 5 and conclusions in the final section 6.

EUVIP 2014, Dec. 10-12, 2014, Paris, France.
978-1-4799-4572-6/14/$31.00 ©2014 IEEE.

2. Related Work

The use of visualization techniques to aid malware analysts is becoming more popular in classifying malware and its behavior. In [**13**] samples are dynamically assessed in a sandbox and their API behavior reports represented as treemaps and thread graphs. The treemaps shows the percentage and type of API calls, file handling or socket creation for example; whereas the thread graph shows a chronological display of executed commands with varying granularity. Nataraj et al [**14**] proposed an approach of classifying malware by transforming each byte value in an executable file into the pixel value of a grayscale bitmap image. They generate texture-based features using GIST [**15**] in high dimensionality space. They have shown this to be very accurate in classifying members of malware families although it is computationally heavy they can achieve an overall accuracy of 97.18%. Han et al [**16**] put forward a technique of classifying malware samples by converting binary files into images and histograms. Classification is performed by comparing histogram similarity between a sample file and a database of known malware. The results presented a false positive rate of 3.5% and a false negative rate of 3%. The studies in [**14**][**16**] conjecture the ability of their methods in detecting packed executables but no conclusive study was performed.

3. Proposed Methodology

In this paper we propose to fuse the techniques of visualization, steganalysis and machine learning into a four stage approach for packed executable detection. The methodology presented in this section accepts an executable file as input and produces a classification result of packed or non-packed as output.

In stage 1 an executable file is converted from .exe format into a bitmap image. Following this the image is filtered to remove noise in stage 2. Singular Value Decomposition (SVD) is used to generate the features in the penultimate stage before classification occurs in the final stage. This process is illustrated in Figure 1. The remainder of this section explains in more detail how the techniques of the individual stages are carried out.

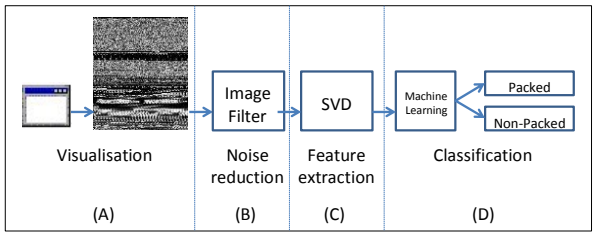

Figure 1 Four stages to portable executable classification

3.1 Visualisation

Following the methodology in [**17**] each byte value 0x00 – 0xFF within the executable file is used as a pixel value in the resultant image. The value 0x00 equates to a fully black pixel, 0xFF to completely white pixel and 254 shades of gray inbetween. Each image has a fixed width of 256 pixels with varying height depending on the file size.

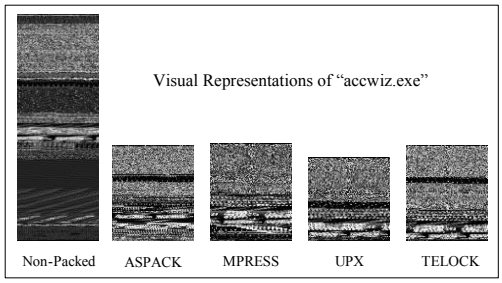

Figure 2 Visualizations of packed and non-packed accwiz.exe

The images displayed in Figure 2 show bitmaps of various versions of the native windows executable '*accwiz.exe*'. The largest image is the non-packed version and the others are obfuscated using the packer named below each one. Figure 2 highlights how a noticeable visual difference can be seen between a non-packed and packed instance. It also demonstrates that although packing algorithms vary across providers, the presence of large portions of encrypted data make packed executables visually similar to one another.

3.2 Noise Reduction

Filtering noise from an image helps to produce a cleaner representation of the data by smoothing over the very fine detail. In natural images it is mostly done for aesthetic purposes but in computer vision it is done to allow more accurate feature generation.

3.3 Feature Extraction

Once the executables are transformed into grayscale bitmap images and pre-processed using the filtering methods they are analysed in the spatial domain using SVD.

3.3.1 Singular Value Decomposition

SVD is popular across signal processing techniques as it is a matrix factorization method which can be used on any matrix comprised of real numbers. It decomposes a matrix A, into the product of a diagonal matrix of non-negative elements, S and two orthogonal matrices U and V. The diagonal matrix S is comprised of non-negative elements which are ordered in decreasing value $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \sigma_{min(m,n)}$ where m and n are the dimensions of A. These values are extracted from S to produce a row vector known as the "*singular value vector*".

3.3.2 Feature Generation

Considering the whole image as a matrix, features are found within groups of pixels in localized areas or sub-matrices of the bitmap. In order to access these features it is proposed that a sliding window of matrix size *N*x*N* will traverse the image generating a *singular value vector* for each unique window. Each row vector contains elements used describe the scaling transformation for each block. The elements are sorted in descending order with the first element being much larger than the last. In order to balance the values for feature generation the natural logarithm of the inverse power of each individual element of the vector is calculated $\ln(\sigma_x^{-1})$. The sum of the elements in the vector is used as the unique value for that individual window and an average of all windows scores is used as the singular value feature for that whole image. A more thorough explanation is outside of the bounds of this work but can be found in [**12**].

3.4 Classification

The extracted features from the SVD analysis are passed to the machine learning tool, WEKA [**18**]. Tenfold crossvalidation is utilized for training and testing the classification model. This method randomly chooses 90% of the total dataset for training and the remaining 10% for testing. This process is executed ten times and the final result is an average of the results from each of the iterations.

4. EXPERIMENTS

The following experiments are carried out to evaluate the suitability and accuracy of using steganalysis to detect packed executable files. A dataset of packed and non-packed files is converted into images and analysed. Three noise filters are individually applied to separate copies of the image file and a set of feature vectors of varying lengths extracted. Each combination of filtering technique and feature vector size is classified using machine learning algorithms to produce the results in section 5.

4.1 Dataset

In order to quantify the performance of utilizing steganalysis techniques to classify packed and non-packed executables, a dataset was created using native Windows XP executable files. 332 non-packed programs were extracted from a fresh install of Windows XP and in turn multiple versions were created using seven different packers. The packing technologies used were comprised of the openly available and popular programs UPX, ASPack, Themida, Obsidium, Telock, RLPack and MPRESS. Obfuscation was attempted for every file with every packer although not all were successful. The total dataset is comprised of 1,667 packed files and 332 non-packed files. In order to balance the dataset each non-packed instance was duplicated 5 times to provide a counterpart for each packed file image. This resulted in 1,676 packed executable images and 1,660 non-packed executable images.

4.2 Selected Noise Reduction Filters

For the filtering stage of the proposed methodology three variations were selected;

1) Wiener filter

Due to its previous success at detecting images embedded with steganographic detail as shown in [**12**] the Wiener filter has been chosen to attempt noise reduction in the following experiments.

2) Median filter

This filter has been shown to be effective at noise removal whilst retaining edge information which can be vital to features within images. With the visualization of executables producing non-natural images, the presence of edges highlight the partitions between separate sections within an executable or recurring patterns in the underlying structure. Retaining these edges could prove critical to generating features which differentiate between a packed and nonpacked file.

3) No filter

For control purposes the third choice of noise reduction was to initiate no filter. Doing so provides a base rate of classification to enable fair comparison of results using different filtering techniques.

4.3 Machine Learning Algorithms

As this is the first time steganalysis techniques have been used to assess packed or non-packed executables the number of features to use for classification is unclear. To address this, a range of feature set sizes is used to evaluate if there is an optimum level for detection. For the proposed approach an analysis of feature vectors of size 1-25 is presented. The sliding window matrix W of size N x N described in the previous section has values ranging from $N = 2$ to $N = 26$.

Classification was performed using WEKA tools and the fully supervised learning algorithms, in particular Support Vector Machine (SVM) and Ibk classifiers. The former is used across multiple disciplines for classification problems including use in the SVD based steganalysis work in [**12**]. The latter, Ibk is an implementation of the k-nearest neighbour algorithm [**19**] which was shown to have strong prediction rates in the classification of visualized malware samples in [**14**] and also performed well during preliminary work in this research.

5. RESULTS

The following figures and tables presents the experimental results and the detection performance of the proposed methodology at classifying packed and non-packed executable files. In contrast to state-of-the-art [3][4][5][6][9], which derives features from domain-knowledge, such as names of sections, entropy scores etc., the proposed approach extracts features by converting

executable files into bitmap images and using an image processing technique popular in steganalysis.

Two machine learning algorithms were used for classification and their overall accuracy scores for each filtering method are presented in Figure 3 and Figure 4.

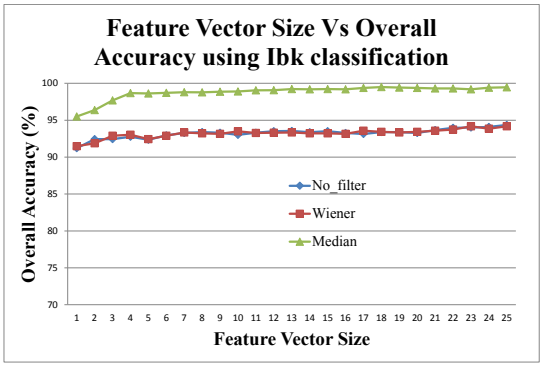

Figure 3 Results from IBK classification

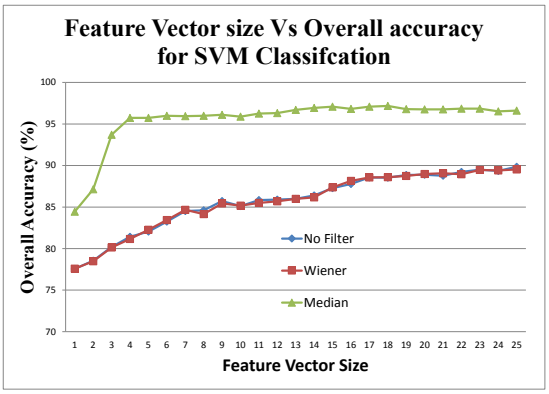

Figure 4 Results from SVM classification

The results from both classifiers show no significant difference between Wiener filtered and non-filtered images. The Median filter, however, significantly improves the classification performance for both classifiers. The ability of the median filter to smooth values and maintain edge information has proved to be an excellent approach for noise reduction. The best classification rate of 99.49% was achieved by using a median filter, a feature vector of 18 values and the Ibk algorithm.

There is a notable difference in the overall accuracies of the machine learning algorithms. Ibk outperformed the SVM in every variation of feature vector size and filtering mechanism. With regards to classification techniques the SVM defines an optimal hyper-plane to divide the instances of data into two separate regions. Whereas the Ibk algorithm makes a decision based on its k^{th} nearest neighbours, for these experiments k=1. The instance, which is being classified, will be given the same class value as its closest neighbour. For all feature vector sizes the Ibk achieves the best classification results, suggest that images which are visually similar are closely grouped in a feature space but not separate enough for a clean hyper-plane division.

The confusion matrix of the top performer as shown in Figure 5 provides the number of correctly and incorrectly classified instances during its classification process. It shows the proposed approach yielded a false positive rate of 0%, false negative rate of 1.01% and an overall accuracy of 99.49%.

		Predicted Class	
		Packed	Non-Packed
Actual Class	Packed	1659	17
	Non-Packed	0	1660

Figure 5 Confusion Matrix for top performer

In order to provide a benchmark, Table 1 provides comparison with related work. It presents the type and number of features required to attain their detection rates. This study's contribution to the field is the ability to classify packed and non-packed executables at a state-of-the-art level with a significantly reduced cost in feature generation. The proposed approach requires less than 10 percent of the total features needed for the currently top ranked research.

Table 1 Comparison with state of the art packed executable detection techniques

Method	Features	Overall Accuracy (%)
Pedrero et al. [**9**]	209 Heuristic	99.80
Shafiq et al. (PE-Miner) [**20**]	189 Heuristic	99.50
Proposed Approach	18 Visual	99.49
Perdisci et al. [**5**]	9 Heuristic	98.91
Han et al. [**3**]	1 Heuristic	97.50
Treadwell et al. [**6**]	8 Heuristic	95.30
Choi et al. (PHAD) [**4**]	8 Heuristic	93.59

6. CONCLUSION

In this paper a novel method of detecting packed executable files using steganalysis was proposed. To the authors' best knowledge, the presented work in this paper is the first ever use of steganalysis and machine learning algorithms for classifying packed and non-packed executables. This study set out to determine if packed executables could be differentiated from non-packed files based on features derived from bitmap representations of the files in question. Experimental studies undertaken revealed that algorithms used for traditional steganalysis are suitable for the detection of packed executables, achieving an overall detection accuracy of greater than 99% and a false positive rate of 0%.

Furthermore, an image based feature extraction has the added benefit of not being susceptible to heuristic based attacks as described in [**11**], because the data is being transformed and utilized in a way that malware authors do not have the means to anticipate.

REFERENCES

[1] H.Pilz M. Morgenstern, Useful and useless statistics about viruses and anti-virus programs, 2010, Proceedings from the CARO Workshop, 2010. http://www.fsecure.com/weblog/archives/Maik_Morgenstern_Statistics.pdf.

[2] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *Security & Privacy*, vol. 5, no. 2, pp. 40-45, March-April 2007.

[3] Seungwon Han, Keungi Lee, and Sangjin Lee, "Packed PE File Detection for Malware Forensics," in *Computer Science and its Applications*, 2009, pp. 1-7.

[4] Yang-seo Choi, Ik-kyun Kim, Jin-tae Oh, and Jae-cheol Ryou, "PE File Header Analysis-Based Packed PE File Detection Technique (PHAD)," in *Computer Science and its Applications, 2008. CSA '08. International Symposium on*, 2008, pp. 28-31.

[5] Andrea Lanzi, Wenke Lee Roberto Perdisci, "Classification of packed executables for accurate computer virus detection," *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1941-1946, October 2008.

[6] Scott Treadwell and Mian Zhou, "A Heuristic Approach for Detection of Obfuscated Malware," in *Intelligence and System Informatics*, Richardson, Texas, 2009, pp. 291-299.

[7] M., S. Tabish, and Muddassar Farooq Shafiq, "PE-probe: leveraging packer detection and structural information to detect malicious portable executables," in *Proceedings of the Virus Bulletin Conference (VB).* , Geneva, 2009.

[8] Chin-Hsuing Wu Tzu-Yen Wang, "Detection of Packed Executables Using Support Vector Machines," in *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, Guilin, 2011, pp. 717-722.

[9] Xabier Ugarte-Pedrero, Igor Santos, Borja Sanz, Carlos Laorden, and Pablo Bringas, "Collective classification for packed executable identification," in *CEAS '11 Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, New York, 2011, pp. 23-30.

[10] Xabier Ugarte-Pedrero, Igor Santos, Borja Sanz, Carlos Laorden, and Pablo Garcia Bringas, "Countering entropy measure attacks on packed software detection," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, Las Vegas, 2012, pp. 164-168.

[11] M. Baig, P. Zavarsky, R. Ruhl, and D. Lindskog, "The study of evasion of packed PE from static detection," in *World Congress on Internet Security (WorldCIS)*, Guelph, ON, 2012, pp. 99-104.

[12] G Gul and Kurugollu F, "SVD-Based Universal Spatial Domain Image Steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 349-353, June 2010.

[13] T. Holz, J. Gobel, and F.C. Freiling, "Visual Analysis of Malware Behaviour Using Treemaps and Thread Graphs," in *6th International Workshop on Visualization for Cyber Security, 2009. VizSec 2009*, New Jersey, USA, 2009, pp. 33-38.

[14] L.Nataraj, S.Karthikeyan, G.Jacob, and B.S. Manjunath, "Malware Images: Visualization and Automatic Classification," in *Vizsec'11*, Pittsburgh, PA, 2011.

[15] A Olvia and A Torralba, "Modelling the shape of a scene: A holistic representation of the satial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145-175, 2001.

[16] K Han, J Lim, B Kang, and E Im, "Malware analysis using visualized images and histograms," *Under review - Springer Journal of Information Security*, 2013.

[17] Lakshmanan, S. Karthikeyan, Gregoire Jacob, and B. S. Manjunath Nataraj, "Malware images: visualization and automatic classification," in *The 8th International Symposium on Visualization for Cyber Security*, Pittsburgh, PA, 2011, pp. 4-10.

[18] University of Waikato, Weka 3 : Data mining software in Java, 2013, A software tool.

[19] David W. Aha, Dennis Kibler, and Marc K. Albert., "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37-66, 1991.

[20] M. Shafiq, S Tabish, F Mirza, and M Farooq, "PE-Miner: mining structural information to detect malicious executables in realtime," in *Recent Advances in Intrusion Detection*, Saint-Malo, France, 2009, pp. 121-141.