2019 2nd International Conference on Data Intelligence and Security (ICDIS)

Robust PDF Malware Detection with Image Visualization and Processing Techniques

Andrew Corum
Department of Computer Science
Calvin College
Grand Rapids, MI 49546
amc49@students.calvin.edu

Donovan Jenkins, Jun Zheng
Department of Computer Science and Engineering
New Mexico Institute of Mining and Technology
Socorro, NM 87801
donovan.jenkins@student.nmt.edu, jun.zheng@nmt.edu

***Abstract*—PDF, as one of most popular document file format, has been frequently utilized as a vector by attackers to covey malware due to its flexible file structure and the ability to embed different kinds of content. In this paper, we propose a new learning-based method to detect PDF malware using image processing and processing techniques. The PDF files are first converted to grayscale images using image visualization techniques. Then various image features representing the distinct visual characteristics of PDF malware and benign PDF files are extracted. Finally, learning algorithms are applied to create the classification models to classify a new PDF file as malicious or benign. The performance of the proposed method was evaluated using Contagio PDF malware dataset. The results show that the proposed method is a viable solution for PDF malware detection. It is also shown that the proposed method is more robust to resist reverse mimicry attacks than the state-of-art learning-based method.**

***Index Terms*—PDF malware, malware detection, image visualization, feature extraction**

I. Introduction

PDF (Portable Document Format) is a file format invented by Adobe for presenting, exchanging and archiving documents that is independent of hardware, software, and operating systems. As one of the most used file formats, PDF documents have become one of the major vectors for malware attacks. This is mainly due to the flexibility of PDF file structure and the ability of embedding different kinds of content such as JavaScript code, encoded streams and image objects etc. These features can be exploited by attackers to embed the malware in PDF files using tools like Metasploit [1], [2]. For example, it was reported that the current popular Ransomware can be hidden inside PDF documents to launch the attacks [3].

Various PDF malware detection techniques have been proposed to address the challenges of PDF malware attacks, including keyword-based techniques, tree-based techniques, code-based techniques and learning-based techniques [1]. An essential step for learning-based PDF malware detection techniques is feature extraction. Two popular kinds features used in the current learning-based techniques are Javascript-based features and structural features [1]. PJScan [4] and Lux0R [5] are two examples that use features related to the embedded Javascript code to detect malicious behavior. Other techniques like PDF Slayer [6], PDFRate [7] and Hidost [8] rely on features extracted from the logical structure of PDF files to detect malicious behavior.

In this paper, we propose a learning-based PDF malware detection method with image visualization and processing, a completely different approach for feature extraction. Using image visualization approaches proposed in [9] and [10], the PDF files are converted to grayscale images first. Then image features can be extracted to represent the distinct visual characteristics of PDF malware and benign PDF files.

The rest of this paper is organized as follows. In Section II, we present the background information about PDF file structure, PDF malware and attacks to evade learning-based PDF malware detection. An overview of the proposed PDF malware detection method with image visualization and processing is presented in Section III. The details of image visualization of PDF files and feature extraction using image processing are introduced in Sections IV and V, respectively. The performance evaluation results of the proposed method are provided in Section VI. Finally, conclusions are drawn in Section VII.

II. Background

A. PDF File Structure

A PDF file contains at least four parts: header, body, cross-reference table, and a trailer. The header contains information on the PDF version used to create the file. The body section contains all the objects the file needs. These objects hold text, images, formatting information, content location, scripts, and more. There are two types of objects: direct and indirect. Indirect objects are typically dictionaries and referenced in the cross-reference table. These objects can call other indirect objects and direct objects. The cross reference table simply lists the names and locations of all the indirect objects. The trailer section contains the name and location of the first object to be rendered (typically called ”nroot”).

B. PDF Malware

There are a number of ways to exploit the PDF file format due to its powerful capability of embedding different kinds of content. For example, a popular attack is to inject malicious Javascript code into a PDF file using specific APIs. The

978-1-7281-2080-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDIS.2019.00024

malicious code can then used to exploit certain vulnerabilities of the PDF reader. Attackers can also embed malicious Shockwave Flash (SWF) files and ActionScript code in a PDF file. This can be used to exploit the vulnerabilities of the Flash interpreter of the PDF reader. Early versions of PDF reader allowed the execution of arbitrary binary files specified in a PDF document which could be exploited by remote attackers to launch malicious attacks as reported in CVE-2010-1240.

C. Evasion of Learning-based PDF Malware Detection

To avoid detection by learning-based approaches, attackers can utilize various attacks to evade the classifiers. Two of these attacks are mimicry [5], [7] and reverse mimicry [11] attacks. The mimicry attacks inject benign PDF objects into a malicious PDF file to cause the malicious PDF file to appear benign. To conduct the mimicry attack, the attacker is aware of the threshold that the classifier uses to distinguish a benign PDF file and a malicious one, and injects benign code until the threshold is crossed. On the contrary, reverse mimicry attacks work by controlling the amount of the malicious code injected into a benign PDF file. The attacker will inject just enough malicious code to achieve their attacking goal, while not crossing the classifier's threshold and be considered as malicious. Figure 1 illustrates the mimicry and reverse mimicry attacks.

III. Overview of Proposed Method

The proposed method for detecting PDF malware with image visualization and processing consists of the following steps as shown in Figure 2:

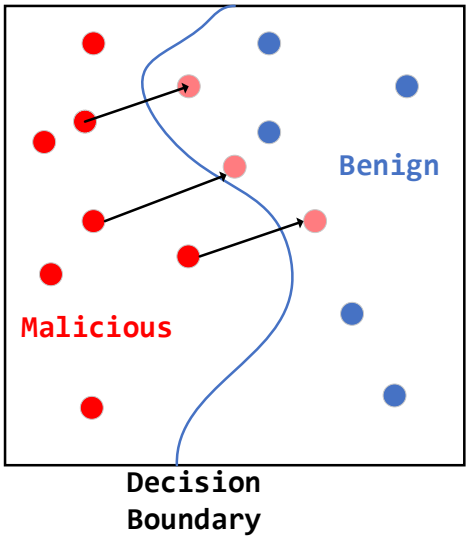
- *Image visualization*: As the preprocessing step, the PDF files are converted to grayscale images using two image visualization techniques, byte plot [9] and Markov plot [10].
- *Image feature extraction*: To characterize the converted grayscale images, image features are extracted which are required as an essential step to build the classification model to classify the PDF files as malicious or benign. In this study, we consider two kinds of images features, keypoint descriptors and texture features.
- *Learning and classification*: By representing each PDF file as a feature vector, a learning algorithm is applied to create a classification model which can be used to classify new PDF files as malicious or benign. Three popular learning algorithms, Random Forests (RF), Decision Trees (DT) and K-Nearest Neighbor (KNN), are adopted in our study.

IV. Image Visualization of PDF Files

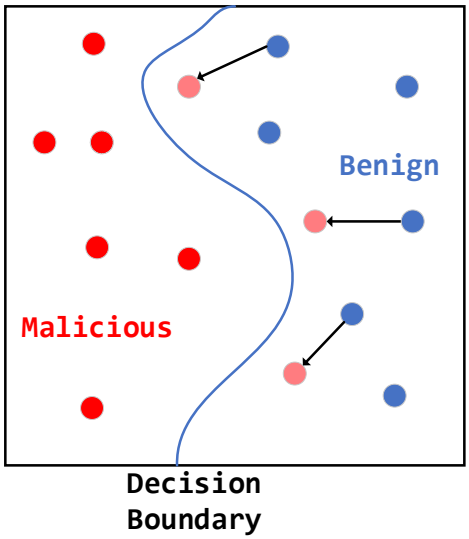
In this section, we introduce how to visualize PDF files as grayscale images using byte plot [9] and Markov plot [10].

A. Byte Plot

Byte plot is a simple and effective method proposed in [9] to visualize malware binaries as grayscale images. To create a byte plot for a PDF file, the PDF binary is read byte by byte sequentially. Each byte is written directly as a pixel in a grayscale image with a range of possible value from 0 to 255. The bytes are placed on the image following scan lines, from left to right. To determine the image size, a method from [9] is used which the image width is determined by the size of the PDF file as shown in Table I. The length of the image is variable which also depends on the PDF file size. Figure 3 shows an example of a byte plot image created from a PDF file.



(a)



(b)

Fig. 1: Illustration of the (a) mimicry and (b) reverse mimicry attacks

B. Markov Plot

The Markov plot is an alternative way to visualize a PDF file as a grayscale image [10]. A Markov plot is created by

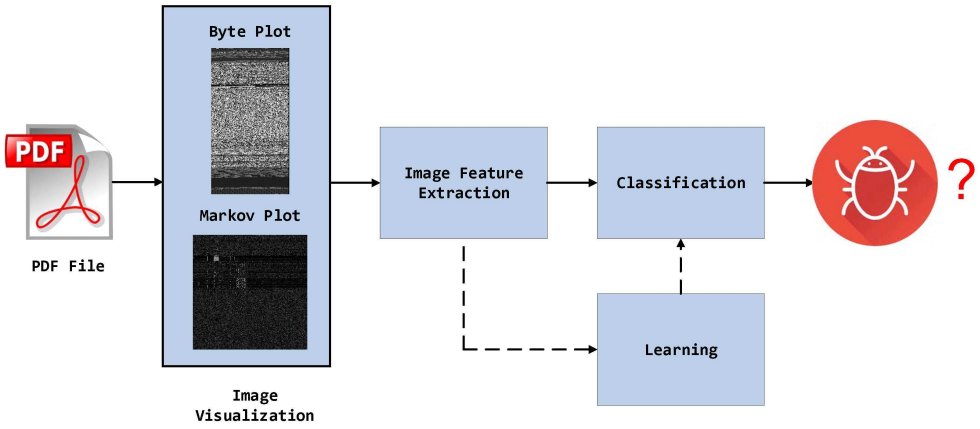


Fig. 2: Proposed PDF malware detetion method with image visualization and processing

TABLE I: Image Width vs. PDF File Size [9]

PDF 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

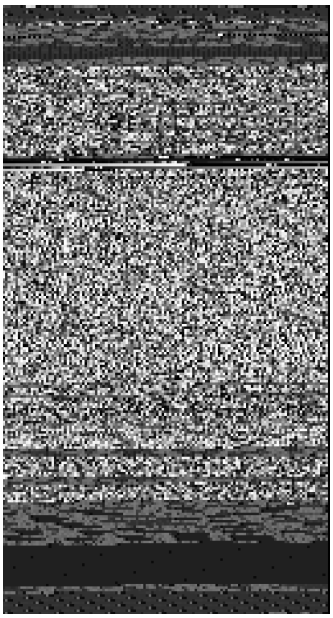


Fig. 3: An example of a byte plot of a PDF file

representing the byte stream of a PDF file as the state transition matrix of a 256-state Markov chain. The state space of the Markov chain is the set of possible values of a byte, $\mathbf{S} = \{0, 1, ..., 255\}$. The state transition matrix $\mathbf{P}$ of the Markov chain is given by $\{P_{i,j}|0 \leq i \leq 255, 0 \leq j \leq 255\}$. $P_{i,j}$ is the transition probability of a byte value i to a byte value j in the PDF byte stream which is calculated as shown in Equation (1).

$$P_{i,j} = \frac{w_{i,j}}{\sum_{k=0}^{255} w_{i,k}} \quad (1)$$

where $w_{i,j}$ is the number of transitions from byte value i to byte value j in the PDF byte stream.

To display as a grayscale bitmap image, the state transition matrix $\mathbf{P}$ is scaled as $\mathbf{I} = \frac{255}{\max(\mathbf{P})}\mathbf{P}$. Note that the image size of Markov plot is the same (256×256) for all PDF files. An example of a PDF Markov plot is shown in Figure 4.

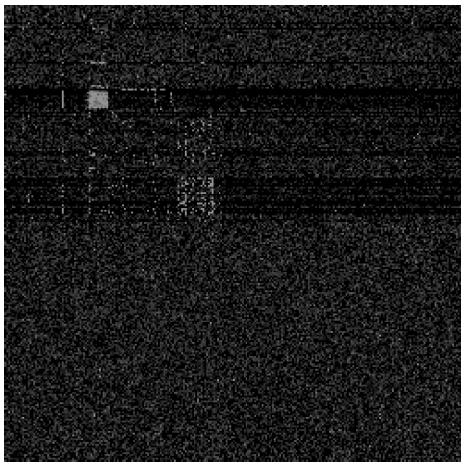


Fig. 4: An example of a Markov plot of a PDF file

V. Feature Extraction using Image Processing

After preprocessing the PDF files using visualization techniques, we extracted features from the images which will be used as input to build the classification models. In this study, we consider two kinds of image features: keypoint descriptors and texture features.

A. Keypoint Descriptors

Keypoint descriptors have been used for object detection and classification. We studied two widely-used keypoint descriptors: Scale Invariant Feature Transform (SIFT) [12] and Oriented FAST and Rotated BRIEF (ORB) [13].

1) SIFT: The SIFT descriptor was developed by Lowe [12] which is widely used for feature extraction in computer vision applications. SIFT can be broken down into four main steps: detection of scale-space extrema, keypoint localization, keypoint orientation, and keypoint description.

First SIFT detects scale-space extrema (local maxima and minima) within the image. These extrema will be starting points for finding the image's keypoints. SIFT algorithm employs Difference of Gaussian (DoG) as a close approximation of Laplacian of Gaussian LoG) to detect the local extrema over scale and space. The potential keypoints found in the first step is then thresholded with a threshold 0.03 [12] to remove low-contrast ones. Since DoG often flags edges as keypints while we are only interested in corners, these unwanted keypoints are removed by setting a threshold for the curvature at an extremum, which is calculated using the Hessian matrix.

After the array of keypoints has been trimmed down, the SIFT algorithm assigns orientations to each keypoint. This is done by applying a histogram of oriented gradients with 36 bins. Finally the keypoint descriptors can be created, using a 16 x 16 neighborhood around each keypoint. This neighborhood is divided up into 16 blocks, and in each block another orientation histogram with 8 bins is taken. Hence a total of 128 bins are available as a descriptor for each keypoint. After this entire process is complete, we are left with a feature vector of size $Num\_Keypoints \times 128$. Figure 5 is an example of keypoints that SIFT identified on the byte plot representation of a PDF.

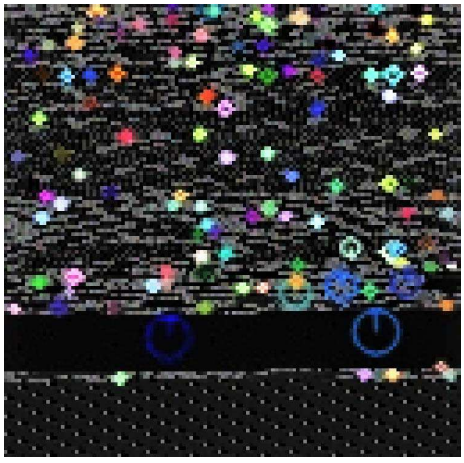


Fig. 5: Keypoints identified by SIFT on a byte plot

2) ORB: The computation of SIFT is slow due to its complexity and descriptor vector length. ORB provides a fast alternative to SIFT [13] which employs the Oriented Features from Accelerated Segment Test (FAST) keypoint detector and the Rotated Binary Robust Independent Elementary Features (BRIEF) descriptor. Due to the significantly slow computation of SIFT and similarity between SIFT and ORB, we only included ORB in the performance evaluation.

FAST was proposed in [14] to find corner keypoints efficiently. ORB appies FAST with a few modifications. To detect keypoints, ORB uses a radius of 9 rather than 3 around the center pixel c. ORB also applies FAST on a multi-scale pyramid of the image and sort the keypoints using the Harris corner measure [15]. The keypoints are then thresholded to obtain the targeted number n points. In our study, n is set to 32. The FAST method does not return any information on corner orientation, so ORB uses the location of the intensity centroid to find the corner's orientation. The intensity centroid is defined in [16] as

$$C = \left(\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}}\right) \tag{2}$$

where the moment $m_{i,j}$ is

$$m_{i,j} = \sum_{p} p_x^i p_y^j I(p) \tag{3}$$

for each pixel $p \in P$ in the corner. Finally, the orientation of the corner is $\theta = \text{atan2}(m_{0,1}, m_{1,0})$.

The BRIEF descriptor was proposed in [17] which is a fast and efficient binary descriptor but without the rotation invariance. BRIEF works by taking a smoothed patch of the image, p. Then this patch is tested at n different paired keypoints found by FAST detector around the image. Each pair of points is tested using Equation (4), where $I(a)$ is the intensity of point a in the smoothed patch p.

$$\tau(p, a, b) = \begin{cases} 1, & I(a) < I(b) \\ 0, & I(a) \geq I(b) \end{cases} \tag{4}$$

The BRIEF feature vector, $\vec{f_n}$, is made up of each n pairs of points tested in this way, as shown in Equation (5).

$$\vec{f_n}(p) = \langle \tau(p, a_0, b_0), \tau(p, a_1, b_1), ..., \tau(p, a_{n-1}, b_{n-1}) \rangle \tag{5}$$

However, in order to account for rotated keypoints, ORB uses a rotated BRIEF (rBRIEF) descriptor [13]. The rBRIEF descriptor is $\vec{g_n}(p, \theta) = \vec{f_n}(p)$ where the paired locations, $a, b \in S_\theta$. The set $S_\theta = R_\theta S$, where S is the matrix of paired locations, $S = \begin{bmatrix} a_0 & a_1 & ... & a_{n-1} \\ b_0 & b_1 & ... & b_{n-1} \end{bmatrix}$, and R_θ is the rotation matrix, $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$. Note that ORB uses a $n = 256$ bits, or 32 bytes, as the size of the BRIEF feature vector, and the keypoint rotation, θ, is only in discrete multiples of $2\pi/30$.

B. Texture Features

Texture features have been widely used to characterize images for various computer vision and pattern recognition applications including image segmentation, biomedical image analysis, object recognition etc. In this study, we consider three methods for extracting texture features from images including local binary patterns (LBP), local entropy, and Gabor filter.

1) LBP: LBP is a typical method used to extract texture features from images [18]. LBP looks at p pixels on the boundary of a bilinear circle with a radius of r around each pixel of the image. LBP compares the intensity of the center pixel c, $I(c)$, with the intensity of each outer pixels $k(k = 1, 2, ..., p)$, $I(k)$. Using the Heaviside step function, LBP checks if the outer pixels have higher or lower intensity than the center pixel. Equation (6) combines each of these comparisons into a single grayscale value, representing the local texture at the center pixel. Once each pixel of the image has been converted in this way, a 64-bin histogram of the image can be made. Each bin of the histogram represents the occurrences of a group of similar local binary textures found throughout the image. These bins are then used as features to build the classification model. In this study, we use a kernel of radius $r = 32$ and $p = 64$ pixels.

$$LBP_{r,p}(c) = \sum_{k=0}^{p-1} H(I(k) - I(c)) \cdot 2^k \tag{6}$$

$$H(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{7}$$

2) Local Entropy: Local entropy is a method proposed in [19] for texture segmentation and thresholding. For a center pixel c, a disk of radius 5 and area A is constructed. The probability assigned to a pixel k with the intensity value $I(k)$, relative to the local neighborhood, is calculated using Equation (8). This probability is then used in a modified version of Shannon's entropy $H(c)$, shown in Equation (9). $H(c)$ is computed for ever pixel in the image, and these values are compiled into a histogram (similar to what was done for LBP in Section V-B1). In this study, we used a histogram with 16 bins which results in a feature vector of size 16.

$$P_k = \frac{I(k)}{A} \tag{8}$$

$$H(c) = -\sum_{k=0}^{A-1} P_k \log_2 P_k \tag{9}$$

3) Gabor Filter: Gabor filters have been used for feature extraction, texture segmentation, and object recognition. Interestingly, these filters behave similarly to the mammalian visual cortex, hence they can be used to recognize textures in an image [20]. In this study, we applied a bank of 24 different Gabor kernels with four different orientations ($\theta = \{0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}\}$), three different standard deviations ($\sigma = \{1, 2, 3\}$), and two different frequencies ($f = c\lambda^{-1} = \{0.05, 0.25\}$). These values define the Gabor kernel, $G_{\lambda,\theta,\sigma}$, as shown in Equation (10).

$$G_{\lambda,\theta,\sigma}(x, y) = e^{-((x'^2+\gamma^2 y'^2)/2\sigma^2)} \cos\left(2\pi \frac{x'}{\lambda} + \phi\right) \tag{10}$$

where

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta, \\ y' &= -x\sin\theta + y\cos\theta, \\ \gamma &= 0.5, \\ \phi &= 0 \end{aligned}$$

After convolving an image with the Gabor kernels, different features can be extracted [20]. In our study, we extracted two features: global mean ($\mu = \frac{\sum_N I(k)}{N}$ for all N pixels in the image) and variance ($\sigma^2 = \frac{\sum_N (I(k)-\mu)^2}{N}$). Using these two measures per filtered image, and an additional 16-bin intensity histogram, a feature vector of size 64 was obtained in our study.

VI. Performance Evaluation

The dataset used for performance evaluation was obtained from Contagio which contains 10,980 malicious and 9,000 benign PDF files [21]. We split the Contagio dataset into two partitions. The first partition consists of 6,000 benign and 6,000 malicious PDF files. This partition was used for testing the performance of different combinations of plot type, image features and classifiers. The second partition consists of the remaining 3,000 benign and 4,980 malicious PDF files, which was used to compare the proposed approach with popular antivirus scanners. The performance metric used for evaluation is F1 score which is calculated using Equation (11), where $Precision$ and $Recall$ are calculated with Equations (12) and (13), respectively.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

$$Precision = \frac{\textit{ture positives}}{\textit{ture positives + false positives}} \tag{12}$$

$$Recall = \frac{\textit{true positives}}{\textit{true positives + false nagetives}} \tag{13}$$

With the first partition of the dataset, we evaluated the performance of different plot type, image features and classifiers using a 10-fold cross-validation. Tables II and III show the results for byte plot and Markov plot, respectively. There are several observations from the results. Obviously, byte plot is a better choice than Markov plot as it always outperforms the latter one. It can seen that keypoint descriptors are not suitable for our task as the performance is much worse compared with that of texture features. The best method is byte plot with local entropy and RF which achieves a F1 score of 0.9924.

TABLE II: 10-fold cross validation results of byte plot

	ORB	**LBP**	**Gabor**	**Local Entropy**
RF	0.8816	0.9869	0.9918	0.9924
DT	0.8821	0.9758	0.9855	0.9874
KNN	0.9188	0.9825	0.9900	0.9851

After training the top-2 performed methods of byte plot or Markov plot with the first partition of dataset, we compare their performance with some popular antivirus scanners using

TABLE III: 10-fold cross validation results of Markov plot

	ORB	LBP	Gabor	Local Entropy
RF	0.8584	0.9766	0.9854	0.9399
DT	0.7022	0.9615	0.9696	0.9025
KNN	0.4477	0.9752	0.9400	0.8987

the second partition of the dataset. The performance of those antivirus scanners was obtained through VirusTotal [22], an online virus scan service with over 70 antivirus scanners. Table IV shows the detection results of our methods and the antivirus scanners. Our best method, byte plot with Gabor Filter and RF, achieves a F1 score of 99.48%. It can be seen that the proposed method has significantly better performance than some popular antivirus scanners like Symantec, Microsoft and nProtect. Some other scanners like McAfee, AVG, Avast achieves slightly better performance than our methods. The results demonstrate that the proposed method is a viable solution for PDF malware detection, especially considering that the Contagio dataset has been available for 7 years, which is plenty enough time to make its way into the databases of those antivirus scanners.

TABLE IV: Performance comparison of the proposed method with popular antivirus scanners

	F1 score
McAfee	0.9979
AVG	0.9964
Avast	0.9959
Byte plot + Gabor + RF	0.9948
Byte plot + Local Entropy + RF	0.9937
Markov plot + Gabor + RF	0.9912
Symantec	0.9898
Markov plot + LBP + RF	0.9736
Microsoft	0.9733
nProtect	0.8887

We also tested our method against reverse mimicry attacks. We obtained a dataset of 1,500 reverse mimicry attacks from [23] which is publicly available. The dataset contains the injections of three different types of contents: embedded executable, embedded JavaScript, and embedded PDF file. Each of the malicious contents was injected into 500 benign PDF files. We compare our method with PDF Slayer [6] which is a state-of-art machine learning-based approach for PDF malware detection. Both methods were trained with the full Contagio dataset and tested with reverse mimicry dataset. The results in terms of the number of detected samples are shown in Table V. Although both methods struggle at detecting reverse mimicry attacks, our method is much more robust than PDF Slayer against the attacks .

VII. Conclusions

PDF is one of the most widely-used document file formats nowadays which has a flexible file structure and can embed different kinds of content such as JavaScript code, encoded streams and image objects etc. This also makes it as a primary target of malware attacks. In this paper, we propose a new learning-based PDF malware detection method which uses image visualization and processing techniques. The PDF files are converted to grayscale images using two image visualization techniques, byte plot and Markov plot. Image features including keypoint descriptors and texture features are then extracted to represent the distinct visual characteristics of the images. The proposed method was evaluated by using a PDF malware dataset from Contagio. Our preliminary results show that the proposed method achieves a good performance compared with popular antivirus scanners, which demonstrate that the method is a viable solution for PDF malware detection. As the security of the learning-based methods are susceptible to evasion attacks, we tested the proposed method against reverse mimicry attacks. It can be seen from the results that the proposed method is more robust than the state-of-art learning-based method in resisting reverse mimicry attacks. In future, the performance of the proposed method can be further improved by investigating more image feature such as wavelet-based texture features [24]. Feature selection can also be applied to select a subset of relevant image features from both keypoint descriptors and texture features to improve the performance. Advanced machine learning algorithms such as deep learning models [25] could also be investigated in future studies.

Acknowledgment

This work was supported by the National Science Foundation under award number CNS-1757945.

References

[1] D. Maiorca and B. Biggio, "Digital investigation of PDF files: unveiling traces of embedded malware," https://arxiv.org/abs/1707.05102.

[2] J. Park and H. Kim, "K-depth mimicry attack to secretly embed shellcode to PDF files," in: Proceedings of International Conference on Information and Science and Applications (ICISA 2017), 2017, pp. 388–395.

[3] https://www.cybersecurity-insiders.com/cyber-attack-with-ransomware-hidden-inside-pdf-documents/

[4] P. Laskov and N. Srndic, "Static detection of malicious javascriptbearin PDF documents," in: Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC 2011), 2011, pp. 373-382.

[5] I. Corona, D. Maiorca, D. Ariu, and G. Giacinto, "Lux0R: Detection of malicious PDF-embedded Javascript code through discriminant analysis of API references," in: Proceedings of the 7th Workshop on Artificial Intelligence and Security (AISec 2014), 2014, pp. 47–57.

[6] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious pdf files detection," in: Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition, 2012, pp. 510–524.

[7] C. Smutz and A. Stavrou, "Malicious PDF detection using meta-data and structural features," in: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012), 2012, pp. 239–248.

[8] N. Srndic and P. Laskov, "Hidost: A static machine-learning based detector of malicious diles," EURASIP Journal of Information Security, Dec. 2016, 2016:22.

[9] L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, "Malware images: Visualization and automatic classification," in: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, 2011, pp. 4:1–4:7.

[10] K. Kancherla, "Image processing techniques for malware detection," Ph.D. thesis, 2015.

[11] D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection," in: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, 2013.

TABLE V: Detection results of the proposed method and PDF Slayer against reverse mimicry attacks

	EXE Embedding	**JS Embedding**	**PDF Embedding**
Byte plot + Gabor + RF	95	176	111
Byte plot + Local entropy + RF	70	162	85
PDF Slayer	52	30	3

[12] D. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, 2004, pp. 91–110.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," In Proceedings of the 2011 International Conference on Computer Vision (ICCV '11), 2011, pp. 2564–2571.

[14] E. Rosten and T. Drummond, "Machine learning for highspeed corner detection," in: Proceedings of the European Conference on Computer Vision, 2006.

[15] C. Harris and M. Stephens, "A combined corner and edge detector," in: Proceedings of Fourth Alvey Vision Conference, 1988, pp. 147–151.

[16] P. Rosin, "Measuring corner properties," Computer Vision and Image Understanding, vol. 73, no. 2, 1999, pp. 291 – 307.

[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in: Proceedings of Computer Vision – ECCV 2010, pp. 778–792.

[18] M. Pietikainen, A. Hadid, G. Zhao, and T. Ahonen, "Local binary patterns for still images," In: Computer Vision Using Local Binary Patterns, Springer, 2011, pp. 13–47.

[19] C. Yan, N. Sang, and T. Zhang, "Local entropy-based transition region extraction and thresholding," Pattern Recognition Letters, vol. 24, no. 16, 2003, pp. 2935 – 2941.

[20] Y. Cho, S. Bae, Y. Jin, K. M. Irick, and V. Narayanan, "Exploring gabor filter implementations for visual cortex modeling on FPGA," In: Proceedings of the 21st International Conference on Field Programmable Logic and Applications, 2011, pp. 311–316.

[21] Contagio, 2013, http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

[22] Virus Total, http://www.virustotal.com/

[23] https://pralab.diee.unica.it/en/pdf-reverse-mimicry/

[24] Y. Huang, V. Bortoli, F. Zhou, and J. Gilles, "Review of wavelet-based unsupervised texture segmentation, advantage of adaptive wavelets," IET Image Processing, vol. 12, no. 9, 2018, pp. 1626–1638.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, Jun. 2017, pp. 84–90.