A Preliminary Study on Malware Classification using Image Pattern

Fauzi Mohd Darus[1], Noor Azurati Ahmad[2], Aswami Fadillah Mohd Ariffin[3]

Faculty of Technology and Informatics Razak, Universiti Teknologi Malaysia
[1]fauzimdarus@gmail.com
[2]azurati@utm.my
[3]aswami@cybersecurity.my

Article history

Received:
23 May 2022

Received in revised form:
2 June 2022

Accepted:
15 June 2022

Published online:
30 June 2022

*Corresponding author
azurati@utm.my

Abstract

Android operating system occupies more than 80% of the world market share in mobile operating system. The popularity of the Android operating system motivates cybercriminals to develop malware targeting this platform. In the first half of the year 2021, there were 1.3 million new malicious Android applications circulated on the globe which the malware analysts need to analyse. Traditional malware analysis techniques are no longer reliable to analyse the huge amount of malware, and they require more resources to process and store them. This research proposed a different approach to analyse Android malware and maintain high classification accuracy with minimal resource usage. 3,900 Android applications consist of malware downloaded from Android Malware Dataset, and benign samples downloaded from APKMirror website were used in this research. The preliminary results of the study show that the image pattern from the same family are analogous meanwhile different family of malware presents distinctive image pattern. Thus, further analysis is needed for different sizes and rotation of extracted malware images.

Keywords: *Malware classification, Embedded Security, Machine Learning, Embedded System, Image Pattern, Visualisation techniques,*

1. Introduction

We live in the digital era where most of our daily activities involve the Internet. Smartphone is the technology in our hands that we use to get connected to the Internet and communicate with our families and friends. With smartphones, we can send chat messages, get the latest updates from online news, or socialize with our friends using social media.

G Data is a company that developed the world's first antivirus software had released a report that shows more than 1.3 million new malicious Android applications were in circulation in the first half of the year 2021 [1]. This indicates that cyber-criminals are eager to target Android devices and they keep on releasing more and more Android malware every day. The Android malware comes in several types and there are trojan, backdoor, worms, botnet and spyware [2].

The rest of the paper is organized as follows: Section 2 summarizes the existing literature related to this study. The data source and methodology are presented in Section 3. This is followed by the results and discussion in Section 4. Finally, Section 5 provides the conclusion of this study.

2. Literature Review

There are two types of analysis that are normally being used by malware analysts; static analysis and dynamic analysis [3]. Static analysis or signature-based analysis is based on specific strings from the disassembled code without executing the binary file. These strings act as the feature and signature of the malware. This analysis can quickly capture the syntax and semantic information for thorough analysis based on the signature database. Even though this technique is fast and accurate, it is easily disturbed if the malware uses code obfuscation and encryption technology. It cannot detect new malware because the feature of the new malware is not found in the signature database [4].

The second type of analysis is dynamic analysis. This technique will analyse the malware behaviour such as network activities, system calls, and file operations by executing the malware in a sandbox. This technique can be used to detect newly created malware because it is not dependent on the malware signature. Unfortunately, this technique has disadvantages where it takes time to execute and affects the system's performance [5].

Naseer et al [6] had summarised issues and challenges in conducting malware analysis. One of the challenges is the limited number of computing and storage resources in malware analysis. Due to the number of new malware that keeps emerging, malware analysts need to continually gather these malwares to keep their malware detection accurate. This activity had cost them money to maintain their machine processing power and storage space resources.

Visualization-based techniques have been introduced to analyse computer malware where the binaries of the malware were translated into images and analysis were done on the images [7]; [5]; [8]; [9]; [10]). The techniques had improved the way to identify and classify malware binaries without the need to do in-depth analysis and achieved more than 90% classification accuracy by using machine learning on the images.

The machine learning algorithms that were used to classify the malware are K-Nearest Neighbours ([11];[12];[8];[9]), Random Forest and Support Vector Machine [13]; [8];[9], XGBoost and LightGBM [14].

Many studies on visualisation-based techniques have been done on computer malware but not many studies on Android malware [15]; [16]. Based on the promising results from computer malware studies, these techniques will be used to identify and classify Android malware.

2.1 Malware Visualisation

Malware visualisation technique was first introduced by L. Nataraj where the computer malware binaries were presented in grey scale images [17]. A byte contains 8 bits (8 binary numbers) where these numbers can be converted into decimal numbers starting from 0b00000000 (0) up to 0b11111111 (255). Each byte represents a grey level where 0 represents black, and 255 is white, while the other values between these two numbers represent different degrees of grey colour.

These bytes values are organised into a two-dimensional matrix and visualised as an image. Based on L. Nataraj., the image's width depends on the size of the binary file while the height is changed accordingly as shown in Table 1.

Table 1:Image width based on file size range

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 – 200kB	384
200 kB – 500kB	512
500 kB – 1000 kB	768
> 1000 kB	1024

By visualising the malware binary into an image, malware analysts can visually view the malware structure and identify the similarity between one image and the other. The example is shown in Figure 1 below, where two malware families show different image patterns. However, when it comes to malware from the same family, the images have the similarities of the same patterns between each malware instance [7].

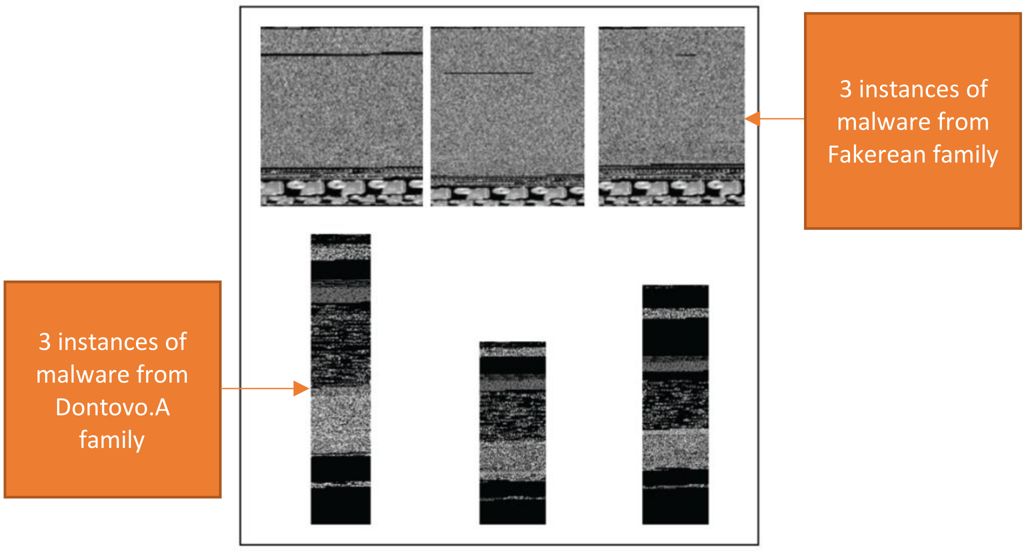

3 instances of malware from Fakerean family
3 instances of malware from Dontovo.A family

Figure 1: Malware visualisation from two different malware families

Another technique was done by [11] where he visualised the Android malware binaries into four different colour formats, i.e. Grayscale, RGB, CMYK and HSL. For the grayscale, the technique he used is similar to [17]. RGB stands for Red (R), Green (G) and Blue (B) is a colour format that is very popular in computer graphics and image processing. Instead of just using 8 bits, RGB will use 24 bits to define a colour from 16 million possible colours.

While for CMYK, the colour format is usually used for printing purposes, and it is almost similar to RGB. It is just that the CMYK represents the colour components for Cyan (C), Magenta (M), Yellow (Y) and Black (K). Finally, for HSL it stands for Hue (H), Saturation (S) and Luminance (L) where hue defines the colour tone, saturation defines the colour tone with grey, and luminance defines the lightness of the colour.

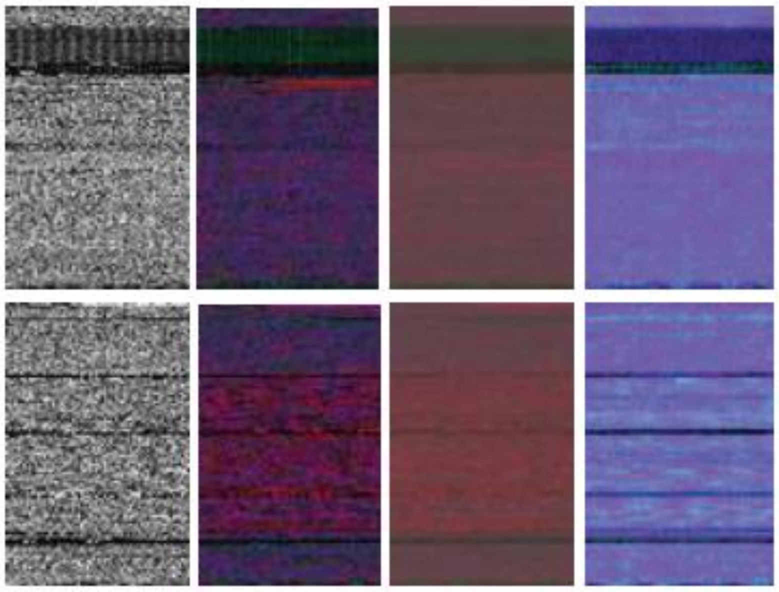


Figure 0: Sample of benign and malware in four image formats (Grayscale, RGB, CMYK and HSL) by [10]

2.2 Feature Extraction

Image processing can be used to help in image segmentation and classification of images. One of the algorithms that can be used in feature extraction is GIST [7]; [15]. GIST contains the scene feature information obtained in a short time, which can be used as a vector to represent the image feature. It has five perceptual properties [18] naturalness, openness, roughness, expansion and ruggedness.

Another feature extraction algorithm is Local Binary Pattern [19]. The image pixels are reorganized into a 3x3 grid and the value at the centre is the threshold for the grids. The threshold value will be compared with the neighbour pixels. If the neighbour's pixel value is greater than the threshold, the value for the neighbour will be '1' and if the neighbour's pixel value is less than the threshold, the value will be '0'.

2.3 Image Classification

Image classification is a proses to identify and classify samples with features similarities from the trained data. The classification can be done by using machine learning algorthims where the features will be trained and tested with the model. Random Forest (RF) is one of the machine learning algorithms that can be used for classification. RF has been used in many types of research related to malware image classification [11]; [15]. RF is an ensemble machine learning where it works by constructing a multitude of Decision Tree during training time and providing the class based on the total votes from the Decision Tree output.

XGBoost is another machine learning that can be used for image classification. XGBoost or eXtreme Gradient Boosting is a scalable machine learning for tree boosting. In 2015, 17 out of 29 machine learning challenges were solved using XGBoost machine learning. It uses less resources, less training time and produces high prediction accuracy [20].

3. Data and Methodology

3.1 Data Source

Related works had been studied, discussed and reviewed to identify the problem related to Android malware analysis. The currently available techniques to visualize and classify Android malware were identified. The type of image to generate, image feature extraction algorithms to be used, and relevant machine learning algorithms to classify the image had been identified in this phase. In this phase, we able to identify how to calculate the accuracy of the classification technique.

Data used for this study are APK files from two categories, malware and benign samples. We have obtained 24,553 Android malware samples from Android Malware Dataset project from Argus Cyber Security [21]. This dataset contains 71 malware families collected from the year 2010 until 2016.

3.2 Image Creation

One classes.dex file will be extracted from each randomly selected APK file. In this phase, all these data sections including the classes.dex will be converted into 8-bit gray scale image.

But before the input file can be converted into an 8-bit image, the script will get the size of the input file. This file size is used to determine the image's width that will be created based on the information in Table 1.

The input file will be read as binary and each 8-bit (1 byte) of the data will be converted into gray scale pixel of the image. If the 8-bit value is 0b00000000 (0), the pixel colour will be black. And if the 8-bit value is 0b11111111 (255), the pixel will be white. While the byte value between these two numbers will represent different degrees of grey colour.

And finally, the images will be resized into 64x64 pixels to ensure all images are in using the same dimension for feature extraction and classification process in the later phase.

3.3 Image Feature Extraction

Computer vision cannot view the image as humans do. If the image has a ‘car’ object, humans can identify the image has tyres, headlamps, door, etc that represent a ‘car’. But for computer, it needs digital information to analyse and translate that data as a ‘car’.

To get that information, the features from the image need to be extracted where the computer will use this information to identify and classify the image accordingly.

GIST descriptor is an image feature extraction that will be used in this experiment. It will extract the information of the scene using low dimensional feature vectors, and it performs well in scene classification.

The image files created from the previous phase will be the input in this workflow. The system will extract the malware family name based on the filename format that we have set in the previous section.

Then the system will extract the features from the given image using the GIST descriptor. The features will then be stored in a dataset based on the family name, section type and orientation.

For example, we have ten images from ‘fusob’ malware family that have been rotated to 90° from ‘data’ section. The system will append all features from these ten images into one single NumPy file and become the dataset for that image group.

This process is repeated until all images have been extracted their features.

4. Results and Discussion

The input for this experiment is the APK files consisting of benign and malware samples. For benign samples, 300 APK files were downloaded from APKMirror website. While for the malware samples, 300 APK files were randomly selected from Android Malware Dataset (AMD), released by Argus Lab.

4.1 APK benign and malware samples

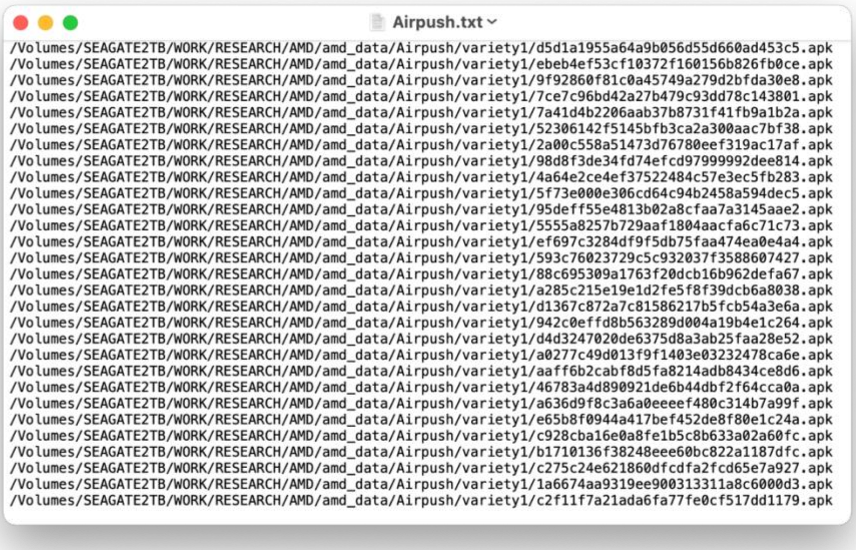
Table 2 shows the number of samples and Android malware families that have been randomly selected for this experiment. These samples consisted of 300 APK files from APKMirror website and 3,600 samples are the malware APK files from 12 different families in Android Malware Dataset.

Table 2: APK files for benign and malware samples

No	Type	Family Name	Source	Number of Samples
1.	Benign	APKMirror	APKMirror	300
2.	Malware	Airpush	AMD	300
3.	Malware	Bankbot	AMD	300
4.	Malware	Dowgin	AMD	300
5.	Malware	Droidkungfu	AMD	300
6.	Malware	Fakeinst	AMD	300
7.	Malware	Fusob	AMD	300
8.	Malware	Jitsu	AMD	300
9.	Malware	Kuguo	AMD	300
10.	Malware	Lotoor	AMD	300
11.	Malware	Mecor	AMD	300
12.	Malware	Rumms	AMD	300
13.	Malware	Youmi	AMD	300
	Total			**3,900**

The randomly selected samples from Android Malware Dataset is recorded in a log file for reference. The log file records the path where the APK files were selected. Some malware families in Android Malware Dataset might have multiple varieties. With this log file, we can easily identify the selected APK files from which variety and family.

Figure 7 shows the APK files that were randomly selected from 'airpush' malware family and all of them came from 'variety1'. As been noticed, Android Malware Dataset used the file's MD5 hash as the APK filename.



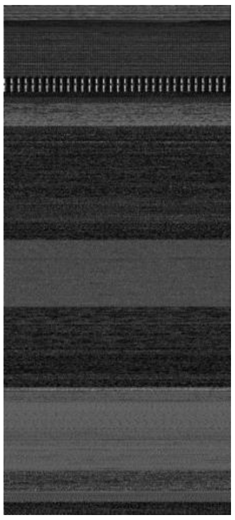
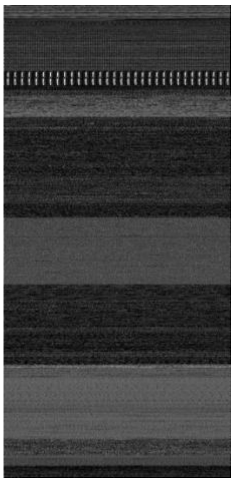
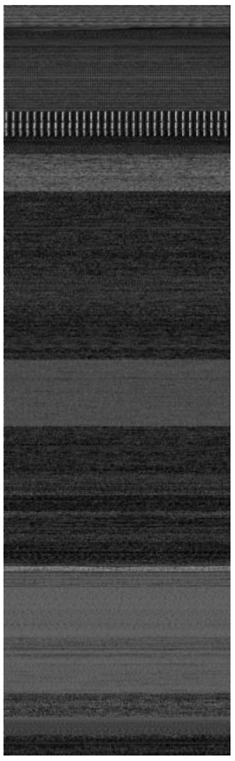
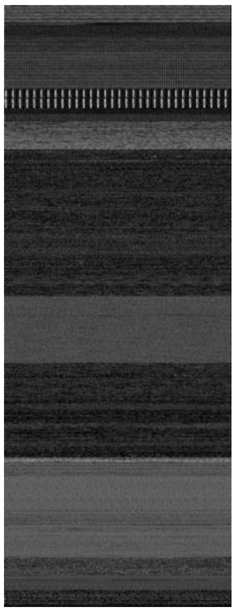
```
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/d5d1a1955a64a9b056d55d660ad453c5.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/ebeb4ef53cf10372f160156b826fb0ce.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/9f92860f81c0a45749a279d2bfda30e8.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/7ce7c96bd42a27b479c93dd78c143801.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/7a41d4b2206aab37b8731f41fb9a1b2a.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/52306142f5145bfb3ca2a300aac7bf38.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/2a00c558a51473d76780eef319ac17af.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/98d8f3de34fd74efcd97999992dee814.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/4a64e2ce4ef37522484c57e3ec5fb283.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/5f73e000e306cd64c94b2458a594dec5.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/95deff55e4813b02a8cfaa7a3145aae2.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/5555a8257b729aaf1804aacfa6c71c73.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/ef697c3284df9f5db75faa474ea0e4a4.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/593c76023729c5c932037f3588607427.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/88c695309a1763f20dcb16b962defa67.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/a285c215e19e1d2fe5f8f39dcb6a8038.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/d1367c872a7c81586217b5fcb54a3e6a.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/942c0effd8b563289d004a19b4e1c264.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/d4d3247020de6375d8a3ab25faa28e52.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/a0277c49d013f9f1403e03232478ca6e.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/aaff6b2cabf8d5fa8214adb8434ce8d6.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/46783a4d890921de6b44dbf2f64cca0a.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/a636d9f8c3a6a0eeeef480c314b7a99f.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/e65b8f0944a417bef452de8f80e1c24a.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/c928cba16e0a8fe1b5c8b633a02a60fc.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/b1710136f38248eee60bc822a1187dfc.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/c275c24e621860dfcdfa2fcd65e7a927.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/1a6674aa9319ee900313311a8c6000d3.apk
/Volumes/SEAGATE2TB/WORK/RESEARCH/AMD/amd_data/Airpush/variety1/c2f11f7a21ada6fa77fe0cf517dd1179.apk
```

Figure 7: List of randomly selected APK files from 'airpush' family

4.2 Android malware images from same family

Table 3 shows four images of Android malware samples from Airpush and FakeInsta families. As you can see, the patterns for these four images are the same for each family. Even though their image lengths are different, but their image patterns are similar. This can be concluded that Android malware from the same family will have the same image pattern.

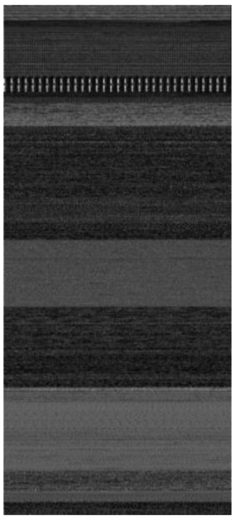
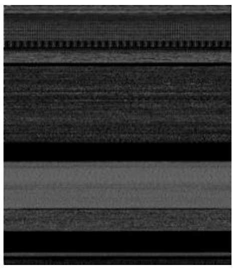
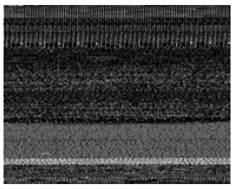
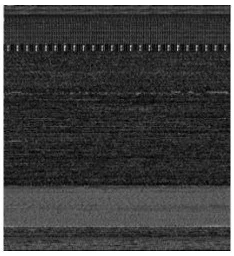
Table 3: Image of malware from same family

Airpush #1	Airpush #2	Airpush #3	Airpush #4
			

4.3 Android malware images from different families

However, when the malware came from different families, the image patterns are different. Table 4 shows the image pattern of malware from airpush, bankbot, dowgin and droidkungfu family. All these images show different image patterns from each other.

Table 4: Image pattern from different malware families

Airpush	Bankbot	Dowgin	Droidkungfu
			

4.4 Resized Android malware images

After the images had been created, they were resized into 64x64 pixel. This process is to standardize the image dimension so that all images are in the equal size for feature extraction and classification.

Table 5 shows four-section images of an airpush malware that have been resized into 64x64 pixels dimension.

Table 0: Resized image of a malware from airpush family

Full	Header	Data	String

5. Conclusion

In conclusion, this study is significant because currently, there is not much research on Android malware using machine learning with visualisation approach. Based on the initial work, we found a lot of research was done on computer malware [7];[5];[8];[9]). These research were successful and produced accurate malware classification results. Using this as motivation, we would like to use the same approach to classify Android malware and to improve the resource usage of the analysis. From this research, we can compare the classification performance with other machine learning algorithms.

Previous research only used classes.dex file and the data section from classes.dex to conduct Android malware classification. This research used strings from classes.dex and converted them into images and used them for Android malware classification with machine learning. The preliminary results of the study show that the image pattern from the same family are analogous meanwhile different family of malware presents distinctive image pattern. Thus, further analysis is needed for different sizes and rotation of extracted malware images.

Acknowledgement

This work has been supported by the grant from UTM and Ministry of Higher Education (MOHE) with the project number R.K130000.7314.4B581.

References

[1] G DATA Software AG (2021) G DATA Mobile Malware Report: Criminals keep up the pace with Android malware, G DATA.

[2] Mahindru, A. and Sangal, A. L. (2021) 'MLDroid—framework for Android malware detection using machine learning techniques', Neural Computing and Applications, 33(10), pp. 5183–5240.

[3] Hadiprakoso, R. B., Kabetta, H. and Buana, I. K. S. (2020) 'Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection', in 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), pp. 8–12.

[4] International Data Corporation (2022) 'Smartphone OS', IDC.

[5] Karimi, A. and Moattar, M. H. (2017) 'Android ransomware detection using reduced opcode sequence and image similarity', in 2017 7th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 229–234.

[6] Naseer, M., Rusdi, J. F., Shanono, N. M., Salam, S., Muslim, Z. Bin, Abu, N. A. and Abadi, I. (2021) 'Malware Detection: Issues and Challenges', Journal of Physics: Conference Series. {IOP} Publishing, 1807(1), p. 12011.

[7] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B. S. (2011) 'Malware Images: Visualization and Automatic Classification', Proceedings of the 8th International Symposium on Visualization for Cyber Security. Pittsburgh, Pennsylvania, USA: ACM, pp. 1–7.

[8] Makandar, A. and Patrot, A. (2017) 'Malware class recognition using image processing techniques', in 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), pp. 76–80.

[9] Manavi, F. and Hamzeh, A. (2017) 'A New Method for Malware Detection Using Opcode Visualization', in 2017 Artificial Intelligence and Signal Processing Conference (AISP), pp. 96–102.

[10] Liu, X., Lin, Y., Li, H. and Zhang, J. (2020) 'A novel method for malware detection on ML-based visualization technique', Computers & Security, 89, p. 101682.

[11] Kumar, A., Sagar, K. P., Kuppusamy, K. S. and Aghila, G. (2016) 'Machine learning based malware classification for Android applications using multimodal image representations', in 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp. 1–6.

[12] Nataraj, L. and Manjunath, B. S. (2016) 'SPAM: Signal Processing to Analyze Malware', IEEE Signal Processing Magazine, 33(2), pp. 105–117.

[13] Nataraj, L. (2015) A Signal Processing Approach to Malware Analysis.

[14] Pan, Q., Tang, W. and Yao, S. (2020) 'The Application of {LightGBM} in Microsoft Malware Detection', Journal of Physics: Conference Series. {IOP} Publishing, 1684(1), p. 12041.

[15] Manzhi, Y. and Qiaoyan, W. (2017) 'Detecting android malware by applying classification techniques on images patterns', in 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 344–347.

[16] Jung, J., Choi, J., Cho, S., Han, S., Park, M. and Hwang, Y.-S. (2018) 'Android malware detection using convolutional neural networks and data section images', in Hung, C.-C. and Said, L. Ben (eds) Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, {RACS} 2018, Honolulu, HI, USA, October 09-12, 2018. ACM, pp. 149–153.

[17] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T. Y. (2017) 'LightGBM: A highly efficient gradient boosting decision tree', in Guyon, I., Luxburg, U. V, Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc., pp. 3147–3155.

[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.

[19] Luo, J.-S., Chia, D. and Lo, -Tien (2017) 'Binary Malware image Classification using Machine Learning with Local Binary Pattern', in 2017 IEEE International Conference on Big Data (Big Data), pp. 4664–4667.

[20] Wang, J., Li, B. and Zeng, Y. (2018) 'XGBoost-Based Android Malware Detection', in Proceedings - 13th International Conference on Computational Intelligence and Security, CIS 2017, pp. 268–272.

[21] Wei, F., Li, Y., Roy, S., Ou, X. and Zhou, W. (2017) 'Deep Ground Truth Analysis of Current Android Malware', in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'17). Bonn, Germany: Springer, pp. 252–276.