Analyzing Image Based Strategies for Android Malware Detection and Classification: An Empirical Exploration

Chirag Jaju[1,*], Dhairya Agrawal[1,*] Rishi Poddar[1,*], Shubh Badjate[1,*], Sidharth Anand[1,*], Barsha Mitra[1,†] and Soumyadeep Dey[2]

[1]*Department of CSIS, BITS Pilani, Hyderabad Campus, Hyderabad, India*

[2]*Microsoft India, India*

Keywords: Android Malware, APK, Image Conversion, CNN, Classification.

Abstract: In recent years, the popularity of Android as a mobile operating system has grown exponentially and so it has been widely used in a huge array of mobile phones. This large scale proliferation of Android has resulted in it being extensively targeted by malware. Numerous families of malware have been developed with the sole purpose of infecting mobile phones and perpetrating different types of attacks on these devices and their users. Naturally, in the past few years, researchers have focused on developing strategies for detecting and classifying malware families. A large number of such strategies are based on converting the malware APK files to grayscale or color images. In this paper, we survey six APK to image conversion techniques and perform a comparative empirical analysis of these methods with respect to malware detection and classification. We implement the six approaches to convert the benign as well as malware binaries into images and then use three CNN-based models to distinguish between benign and malware files and also to classify the various malware families. We use two very popular open-source Android malware datasets, CICAndMal2017 and the Drebin dataset for comparing the performance of the different image conversion techniques for the detection and classification tasks in terms of accuracy and F1-score. The results of the study provide insights into the relative performance of these approaches and help to determine the combination of the image conversion approach and the classification model that provides the best detection and classification performance.

1 INTRODUCTION

Mobile phones, especially smartphones have become an indispensable aspect of our lives. These smartphones have several types of applications installed on them and store a huge amount of data, a large percentage of which is user data and thus is quite sensitive in nature. As a result, smartphones are targeted by hackers and cyber criminals for stealing private user information and for perpetrating various malicious activities exploiting such information. These attacks are primarily launched by infecting the smartphones with malicious softwares or malwares. Since Android is one of the most widely used mobile operating systems, in recent years, scores of Android malware have been developed and disseminated into the cyberspace. This has triggered the design and development of various malware detection and classification strategies as countermeasures. In the recent past, researchers have focused on creating image based malware detection and classification methods which include (Ni et al., 2018), (Gibert et al., 2020), (Mohammed et al., 2021), (O'Shaughnessy and Sheridan, 2022), (Joyce et al., 2023). These approaches convert the malware APKs into images (grayscale or color) and identify the malware families by classifying these converted images. Such methods differ from one another with respect to the algorithm for creating the images.

In this paper, we study six APK to image conversion techniques and compare as well as analyze their performance. The methods that we have selected include (Kumar et al., 2016), (Kalash et al., 2018), (Fang et al., 2020), (Ünver and Bakour, 2020), (Zhang et al., 2021) and (Zhu et al., 2023). We have implemented the image conversion methods proposed in these papers to obtain either grayscale or color images from the APKs. The converted images are

[*]Chirag Jaju, Dhairya Agrawal, Rishi Poddar, Shubh Badjate and Sidharth Anand have equal contribution

[†]Corresponding Author

Jaju, C., Agrawal, D., Poddar, R., Badjate, S., Anand, S., Mitra, B. and Dey, S.
Analyzing Image Based Strategies for Android Malware Detection and Classification: An Empirical Exploration.
DOI: 10.5220/0012139100003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 863-869
ISBN: 978-989-758-666-8; ISSN: 2184-7711

then classified using Convolutional Neural Network (CNN) based models like Resnet50 (He et al., 2015), MobileNetV2 (Sandler et al., 2019), and 3C2D (Mohammed et al., 2021). The primary objective of our work is to provide an experimental comparative analysis of the above-mentioned image conversion based malware detection and categorization techniques and present an insight regarding the relative performance of these approaches to researchers exploring this domain. The insights obtained from our empirical study will help to determine the combination of the image conversion approach and the classification model that provides the best detection as well as classification performance. It is to be noted here that we focus only on the algorithms to obtain the converted images presented in the above-mentioned works in an attempt to study and analyze how the performance of malware detection and categorization is affected by various image transformation techniques. Our study does not focus on presenting any new neural network architecture or classification method for malware detection and identification.

The rest of the paper is organized as follows. Section 2 outlines the preliminaries related to APK files. In Section 3, we describe the different techniques to convert the APK files to grayscale and color images. Section 4 presents detailed information regarding the datasets used for the experiments. Experimental results and their analysis are presented in Section 5 with Section 6 concluding the paper.

2 APK PRELIMINARIES

An APK file is an Android application package, which is used to distribute and install applications on devices running the Android operating system. It is essentially a compressed archive of all the files that make up the application. The APK file contains the compiled code, resources, and manifest files that are necessary for the application to run properly. The contents of an APK file typically include the following components:

- **AndroidManifest.xml.** This is an XML file that contains information about the application, such as the package name, version number and necessary permissions. It also contains information about the application's activities, services, broadcast receivers, and content providers.
- **resources.arsc.** This folder contains all the resources used by the application such as images, audio, video, strings, and layouts.
- **classes.dex.** This file format is used for storing compiled code for Android applications. It contains the Dalvik bytecode, which is the bytecode that Android devices use to run applications. The Dalvik bytecode is compiled from Java class files and is optimized for Android devices. The .dex file format consists of several distinct components:
  - *Header*: contains the header information which includes the file size, the number of strings, types, classes, and methods, and the checksum.
  - *Strings*: contains the strings used in the application.
  - *Types*: contains information about the types of data used in the application.
  - *Classes*: contains information about the classes that are used in the application.
  - *Methods*: contains information about the methods that are used in the application.
  - *Data*: contains the actual data used by the application.
  - *Debug Info*: contains debugging information which can be used to debug the application.
- **lib.** This folder contains any native libraries used by the application.
- **Assets.** This folder contains any additional assets needed by the application, such as text files, databases and other files.
- **META-INF.** This is the directory that contains the cryptographic signature of the application and is used to verify the authenticity of the application.

3 IMAGE CONVERSION TECHNIQUES STUDIED

In this section, we first describe the method for transforming binary files to images and then briefly discuss the techniques that we have considered in this work for converting the malware APKs to images.

3.1 Byteplot-based Binary File to Image Conversion for APKs

The process of converting binary files to images is known as *byteplot-based binary file to image conversion*. This process is a powerful tool for reverse engineering binary files and allows for the visualization of their internal static structure. The byteplot-based binary file to image conversion method was first introduced by (Conti et al., 2008). The authors present an approach for transforming binary files into visual images that are referred to as byteplots. This method provides a visual representation of binary file fragments to help in the identification of common file formats, thus improving the capabilities of text-based hex editors. Byteplot-based binary file to image conversion involves converting a binary file into a 1D array containing 8-bit unsigned integers. (Nataraj et al., 2011) applied byteplots for the first time to represent malware binaries as images for the purpose of malware classification.

This paper explores the use of byteplot-based binary file to 1D array of unsigned integers conversion strategy for converting APK files to 1D arrays of unsigned integers. Subsequently, these arrays are used to obtain the image representations of the APKs. Six techniques are explored for this purpose, including three grayscale and three color image conversion methods. The converted images are then used for various downstream tasks such as benign vs. malware classification and malware family classification. In the following sub-sections, we describe the methods considered in this study for converting APKs to images.

3.1.1 Converting APKs to Grayscale Images

Converting a binary file into an image can be done using byteplot. The resultant 1D array can be considered as a grayscale image with a range of 0 to 255 (0 corresponding to black and 255 corresponding to white). The width of the image is fixed, while the height can vary across different file sizes as mentioned in Table 1. The information presented in the tables is obtained from (Nataraj et al., 2011).

Table 1: Relation between File Size and Image width (Nataraj et al., 2011).

File Size (in kB)	**Width**
< 10	32
10 – 30	64
30 – 60	128
60 – 100	256
100 – 200	384
200 – 500	512
500 – 1000	768
> 1000	1024

Based on the above-mentioned strategy to convert binary files to grayscale images, we have explored three approaches that are discussed next. We refer to the methods as *M1Gray*, *M2Gray* and *M3Gray*.

- **M1Gray.** In this method, the entire APK file is considered as a binary file, and then the byteplot-based technique is utilized to represent the APK as an image file (Kalash et al., 2018).
- **M2Gray.** The AndroidManifest.xml and classes.dex files have been observed to contain enough information to classify various malware families (Zhang et al., 2021). As a result, these two files have been used to convert an APK to an image for the purpose of malware classification. The AndroidManifest.xml file describes the behavior of the application and how it will be executed, while the classes.dex code is the compiled form of the application, containing the main code.
- **M3Gray.** Researchers have investigated the use of resources.arsc files, along with AndroidManifest.xml, and classes.dex files for malware classification (Ünver and Bakour, 2020). Resources.arsc files contain binary version of the compiled resources of an Android application as well as important information about user interface layouts.

3.1.2 Converting APKs to Color Images

In this paper, we have investigated three methods for visualizing an APK as a color image for malware identification. The approaches explored in this work are referred to as *M1Color*, *M2Color* and *M3Color* and are discussed below.

- **M1Color.** The AndroidManifest.xml file and the data part of the classes.dex file are initially used to create a 1D array consisting of unsigned integers as per the method described in Sub-section 3.1. This 1D array is then converted to a 2D array using the width information from Table 1. Finally, the 2D array is reshaped into a 3D array, which can be used to construct an RGB image, providing a visual representation of the data (Kumar et al., 2016).
- **M2Color.** Researchers have observed that the Index region of the classes.dex file contains enough information for effective malware classification. To reduce redundancy and facilitate analysis, the .dex file is pre-processed by removing the header and data sections, leaving only the Index section (Zhu et al., 2023). The six hexadecimal characters are then converted to three channels of numbers between 0 and 255 and stored in a 2D array. This array is reshaped to a 3D array to construct an RGB image.
- **M3Color.** Using information from different sections of a .dex file, an APK to a color image conversion technique is presented in (Fang et al., 2020). The header section contains the byte offset of each section, and this information is used to calculate the RGB channels for that section.

To calculate the green channel, the binary format of the .dex file is used to build a vector consisting of zeros and ones, which is then transformed into an $n \times m$ matrix. The entropy matrix, which represents the red channel of the image, is calculated using Eq 1, where c_i represents the frequency of byte i and $p(ci)$ is its probability. The entropy is then mapped to the range [0 - 255] using Eq 2. The blue channel represents the proportion of each section in the .dex file, which is calculated and converted to the range [0 - 255] using Eq 3.

$$\text{entropy} = -\sum_{i=0}^{255} p(c_i) \log_2 p(c_i) \quad (1)$$

$$\text{R} = (\text{entropy}^2 \mod 8) \times \frac{255}{8} \quad (2)$$

$$\text{B} = \frac{\text{SectionSize}}{\text{FileSize}} \times 255 \quad (3)$$

4 DATASET DESCRIPTION

In this work, we have used two datasets for the malware detection and classification tasks - the Drebin dataset (Arp et al., 2014) and CICAndMal2017 (Lashkari et al., 2018). In the following subsections, we provide a detailed description of each dataset.

4.1 Drebin Dataset

The Drebin dataset is a publicly available resource for classification of Android malware families. It contains 1,29,013 Android applications, with 5,560 of them being malicious. The dataset was collected over a period of more than two years, from August 2010 to October 2012. The samples have been collected from Google Play Store, different alternative Chinese and Russian Markets as well as from other sources like Android websites, security blogs and malware forums. Moreover, all samples from the Android Malware Genome Project (Spreitzenbarth et al., 2013), (Arp et al., 2014) are present in the dataset.

The dataset is divided into two categories - benign applications (Google Play Store - 96,150 applications, different alternative Chinese Markets - 19,545 applications, alternative Russian Markets - 2,810 applications, and other sources - 13,106 applications) and malicious applications (detected by at least two of the ten anti-virus scanners used which include ClamAV, AVG, F-Secure, Kaspersky, AntiVir, McAfee, Panda, BitDefender, Sophos and ESET).

The Drebin dataset comprises of 179 different malware families, including several families that are in active distribution in current application markets. Drebin is one of the largest and most widely used datasets for Android malware classification and is a valuable resource for researchers and practitioners to test and develop their malware identification algorithms. The dataset contains a high degree of imbalance with respect to the number of malware instances per family. To address this issue, we have limited our experiments to the top 20 malware families present in the dataset. These top 20 families are shown in Table 2.

Table 2: Top 20 Families of Malware of the Drebin dataset (Arp et al., 2014).

Family	Number	Family	Number
FakeInstaller	925	Adrd	91
DroidKungFu	667	DroidDream	81
Plankton	625	LinuxLotoor	70
Opfake	613	GoldDream	69
GingerMaster	339	MobileTx	69
BaseBridge	330	FakeRun	61
Iconosys	152	SendPay	59
Kmin	147	Gappusin	58
FakeDoc	132	Imlog	43
Geinimi	92	SMSreg	41

4.2 CICAndMal2017

CICAndMal2017 is an Android malware dataset that consists of over 10,854 samples (4,354 malware and 6,500 benign) obtained from multiple sources (Lashkari et al., 2018). The malware samples are classified into four categories: Ransomware, Adware, SMS Malware and Scareware. The Ransomware category consists of samples from the following malware families - (i) Charger, (ii) Jisut, (iii) Koler, (iv) LockerPin, (v) Simplocker, (vi) Pletor, (vii) PornDroid, (viii) RansomBO, (ix) WannaLocker and (x) Svpeng. The Adware category consists of samples of (i) Dowgin, (ii) Ewind, (iii) Feiwo, (iv) Gooligan, (v) Kemoge, (vi) koodous, (vii) Mobidash, (viii) Selfmite, (ix) Shuanet, and (x) Youmi families. The Scareware category consists of the following malware families - (i) AndroidDefender, (ii) AndroidSpy.277, (iii) AV for Android, (iv) AVpass, (v) FakeApp, (vi) FakeApp.AL, (vii) FakeAV, (viii) FakeJobOffer, (ix) FakeTaoBao, (x) Penetho and (xi) VirusShield. Lastly, the SMS Malware category consists of samples from (i) BeanBot, (ii) Biige, (iii) FakeInst, (iv) FakeMart, (v) FakeNotify, (vi) Jifake, (vii) Mazarbot, (viii) Nandrobox, (ix) Plankton, (x) SMSsniffer and (xi) Zsone families. The samples have been collected from Google play published in three consecutive years - 2015, 2016, and 2017.

Table 3: Experimental Results for CICAndMal2017 for Benign vs. Malware Classification.

Model	**M1Gray**		**M2Gray**		**M3Gray**		**M1Color**		**M2Color**		**M3Color**	
	Acc	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1
Resnet50	79.90	80.02	82.84	82.86	79.90	79.92	81.86	81.65	78.92	79.04	84.80	84.89
MobileNetV2	88.24	88.17	88.24	88.22	**89.22**	**89.23**	85.78	85.65	84.80	84.48	84.80	84.87
3C2D	85.78	85.72	86.76	86.79	86.27	86.27	82.84	82.83	85.29	85.33	83.82	83.84

5 RESULTS AND DISCUSSION

In this section, we discuss our experimental setup, the experimental results and an analysis of the results obtained. We have used Convolutional Neural Network (CNN) based architectures such as Resnet50 (He et al., 2015), MobileNetV2 (Sandler et al., 2019), and a simple but efficient shallow CNN-based architecture 3C2D (Mohammed et al., 2021) for our experiments. We have evaluated the performance of the methods with respect to the ability to distinguish between benign and malware data using the benign data present in the CICAndMal2017 dataset. We have also evaluated the performance of the six APK to image transformation methods on the CICAndMal2017 dataset and the Drebin dataset with the top 20 malware families. For classification purposes, each converted image is resized to a 256×256 dimension image before feeding it to the CNN architectures. We have compared the performance in terms of the popular classification metrics Accuracy (*Acc*) and F1-Score (F_1) to show the effectiveness of each of the techniques.

The overall comparison of the strategies for the benign vs. malware classification on the CICAndMal2017 dataset, malware family classification on the CICAndMal2017 dataset and the Drebin dataset are presented in Tables 3, 4, and 5 respectively. In these tables, the red colored cells indicate the best performance of a particular method across the three CNN-based models, the gray colored cells correspond to the best performance of a specific model across the six methods and the bold-faced cells indicate the best overall performance across all methods and models. For the overall best performance, the cells are colored gray and the values are also written in bold-face (the red color has not been shown). We have used only the CICAndMal2017 for classifying benign and malware files because out of the two datasets considered in this work, only CICAndMal2017 contains benign APK samples. The Drebin dataset contains the extracted features from the benign APKs and hence is not suitable for our empirical study.

Results for the benign vs. malware classification depicted in Table 3 provide the following insights:

- Resnet50 model achieved the highest accuracy of 84.8% and the highest F1-score of 84.89% for the M3Color conversion technique. The model achieved the lowest accuracy of 79.9% and lowest F1-score of 79.92% when used for the M2Gray conversion technique.
- MobileNetV2 achieved the highest accuracy of 89.22% and the highest F1-score of 89.23% while using the M3Gray method. The model achieved the lowest accuracy of 84.8% and lowest F1-score of 84.48% when used for the M2Color approach.
- 3C2D gave the highest accuracy of 86.76% and the highest F1-score of 86.79% for M2Gray, whereas achieved the lowest accuracy of 82.84% and the lowest F1-score of 82.83% for M1Color.
- Overall, MobileNetV2 provided the highest accuracy and F1-score values when used in conjunction with M3Gray conversion technique. However, all three classification models gave, on an average, an accuracy and F1-score of more than 80% for all six conversion methods.

We next analyze the results of malware categorization. Some observations from Table 4 for CICAndMal2017 are as follows:

- Resnet50 model achieved the highest accuracy and F1-score for the M1Color method, with 54.76% accuracy and 54.95% F1-score.
- MobileNetV2 achieved the maximum accuracy and F1-score for M3Gray, with 54.76% accuracy and 55.38% F1-score.
- 3C2D model achieved the highest accuracy and F1-score for M1Gray method, with 61.90% accuracy and 62.47% F1-score.
- Overall, the results demonstrate that 3C2D is the most accurate and performs the best for the M3Color method, while the MobileNetV2 model performs the best for M2Gray. MobileNetV2 gives the best performance for M3Gray, and its performance is more or less consistent across all the conversion techniques. Resnet50 gives the best performance for the color-based conversion methods.

Table 5 provides the performance of the three classification models for the six APK to image conversion approaches for top 5, top 10 and top 20 classes of the

Table 4: Experimental Results for CICAndMal2017.

Model	**M1Gray**		**M2Gray**		**M3Gray**		**M1Color**		**M2Color**		**M3Color**	
	Acc	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1
Resnet50	42.86	43.39	39.29	39.22	53.57	53.80	54.76	54.95	53.57	53.32	51.19	51.92
MobileNetV2	47.62	48.27	51.19	50.70	54.76	55.38	46.43	47.43	50.00	50.83	50.00	50.59
3C2D	**61.90**	**62.47**	48.81	49.20	59.52	60.56	59.52	59.94	59.52	60.23	55.95	56.87

Table 5: Experimental Results for the Top 5, Top 10 and Top 20 Malware Families of the Drebin Dataset.

Dataset	**Model**	**M1Gray**		**M2Gray**		**M3Gray**		**M1Color**		**M2Color**		**M3Color**	
		Acc	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1	*Acc*	F_1
Top5	Resnet50	93.36	93.25	92.73	92.80	92.89	93.21	91.47	91.61	88.78	88.76	93.05	93.06
	MobileNetV2	91.47	91.64	92.73	92.79	92.42	92.53	90.84	90.93	88.47	88.31	90.52	90.31
	3C2D	92.42	92.40	93.52	93.49	91.94	92.00	93.52	93.52	90.36	90.48	**93.84**	**93.85**
Top10	Resnet50	80.31	76.26	91.50	89.62	91.30	90.52	91.91	90.93	83.75	80.81	76.87	71.59
	MobileNetV2	76.20	69.64	81.05	76.02	84.69	81.70	75.46	69.91	78.35	73.34	79.97	75.36
	3C2D	**97.71**	**97.69**	97.24	97.23	97.10	97.07	96.70	96.63	95.08	95.01	96.22	96.15
Top20	Resnet50	74.66	66.89	51.23	37.51	80.95	75.88	74.66	67.23	69.62	61.02	71.42	63.01
	MobileNetV2	65.79	54.50	73.58	65.51	63.51	53.52	64.41	56.12	54.34	43.52	65.43	55.12
	3C2D	**96.29**	**96.28**	96.29	96.20	96.29	96.23	94.97	94.83	93.65	93.56	93.53	93.20

Drebin dataset. The following can be observed for the top 5 classes from the table.

- For top 5 classes, the Resnet50 model performs the best with an accuracy of 93.36% and an F1-score of 93.25% for the conversion method M1Gray.
- MobileNetV2 gives the best performance with M2Gray which has an accuracy of 91.47% and an F1-score of 91.64%.
- 3C2D performs the best among all the three classifiers giving an accuracy of 93.84% and an F1-score of 93.85% for the M3Color method.
- All three models with all six APK to image conversion techniques perform similarly for Drebin top 5 classes.

The observations for the top 10 classes of Drebin from Table 5 are:

- The best performance of Resnet50 has an accuracy of 91.91% and an F1-score of 90.93% for the method M1Color.
- For the M3Gray method, MobileNetV2 has the best accuracy of 84.69% and the highest F1-score of 81.70%.
- 3C2D performs the best with an accuracy of 97.71% and an F1-score of 97.69% with M1Gray.

Table 5 depicts the following for the top 20 classes.

- The best performance of the Resnet50 model is obtained for the conversion method M3Gray with an accuracy of 80.95% and an F1-score of 75.88%.
- The best performance of MobileNetV2 provides an accuracy of 73.58% and an F1-score of 65.51% with the M2Gray conversion technique.
- 3C2D model has 96.29% accuracy and 96.28% F1-score for M1Gray. These metric values are the highest across all the methods.

Overall, the performance of the models varies for the three different dataset classes with the 3C2D model performing the best for all the three dataset classes, in terms of the highest values of accuracy and F1-score. Resnet50 gives the second best performance, followed by MobileNetV2. It can be observed from Table 5 that the performance of the 3C2D model increases significantly with the increase in the number of classes and the imbalance in the classes compared to that of Resnet50 and MobileNetV2. Thus, it can be said that 3C2D is capable of handling class imbalance existing in the malware dataset effectively. This is a vital aspect of classification performance since in the real-world, it is not necessary that the same number of samples will be available for all types of malware. Moreover, it has been observed through experiments that the simple APK to image conversion technique M1Gray, which uses the byteplot method to convert the entire APK file into a grayscale image without any parsing of the individual files present in the APK, is effective for malware classification.

3C2D is a shallow network composed of three convolutional layers followed by two fully connected layers. Each convolutional layer is accompanied by a max pooling layer, resulting in a reduced input image size of $\frac{1}{3}$ before entering the fully connected layers. This allows the model to learn long-range dependencies present in the embedded feature space. We are of the opinion that this fact contributes towards the improved performance of 3C2D compared to other classification models. However, the lesser degree of image size reduction before the fully connected layers leads to a large number of parameters being associated with the model, resulting in a larger model size for 3C2D.

6 CONCLUSION

In this paper, we have empirically analyzed the performance of image based malware detection and classification techniques. Each approach converts an APK to either a grayscale or a color image. These converted images are then input to CNN-based models like Resnet50, MobileNetV2, and 3C2D to distinguish between benign and malware samples as well as identify the various malware families. We have used two widely used open-source datasets, CICAndMal2017 and the Drebin dataset for our experiments. Our experimental results show that 3C2D is capable of providing the most accurate performance for grayscale-based techniques. Based on the observations and insights obtained from this work, in future, we intend to design lightweight malware detection and categorization strategies suitable for resource constrained environments like mobile devices. Moreover, we wish to explore different non-image based features as well for malware classification.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the computing time provided on the high performance computing facility, Sharanga, at the Birla Institute of Technology and Science - Pilani, Hyderabad Campus.

REFERENCES

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., and Rieck, K. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*. The Internet Society.

Conti, G., Dean, E., Sinda, M., and Sangster, B. (2008). Visual reverse engineering of binary and data files. In *International Workshop on Visualization for Computer Security*, page 1 – 17.

Fang, Y., Gao, Y., Jing, F., and Zhang, L. (2020). Android malware familial classification based on dex file section features. *IEEE Access*, 8:10614–10627.

Gibert, D., Mateu, C., and Planes, J. (2020). Hydra: A multimodal deep learning framework for malware classification. *Computers & Security*, 95:101873.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Joyce, R. J., Amlani, D., Nicholas, C., and Raff, E. (2023). Motif: A malware reference dataset with ground truth family labels. *Computers & Security*, 124:102921.

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D. B., Wang, Y., and Iqbal, F. (2018). Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

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)*, pages 1–6.

Lashkari, A. H., Kadir, A. F. A., Taheri, L., and Ghorbani, A. A. (2018). Toward developing a systematic approach to generate benchmark android malware datasets and classification. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–7.

Mohammed, T. M., Nataraj, L., Chikkagoudar, S., Chandrasekaran, S., and Manjunath, B. (2021). Malware detection using frequency domain-based image visualization and deep learning. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 7132.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. (2011). Malware images: Visualization and automatic classification.

Ni, S., Qian, Q., and Zhang, R. (2018). Malware identification using visualization images and deep learning. *Computers & Security*, 77:871–885.

O'Shaughnessy, S. and Sheridan, S. (2022). Image-based malware classification hybrid framework based on space-filling curves. *Computers & Security*, 116:102660.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). Mobilenetv2: Inverted residuals and linear bottlenecks.

Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., and Hoffmann, J. (2013). Mobile-sandbox: Having a deeper look into android applications. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, page 1808–1815, New York, NY, USA. Association for Computing Machinery.

Zhang, W., Luktarhan, N., Ding, C., and Lu, B. (2021). Android malware detection using tcn with bytecode image. *Symmetry*, 13(7).

Zhu, H., Wei, H., Wang, L., Xu, Z., and Sheng, V. S. (2023). An effective end-to-end android malware detection method. *Expert Systems with Applications*, 218:119593.

Ünver, H. and Bakour, K. (2020). Android malware detection based on image-based features and machine learning techniques. *SN Applied Sciences*, 2(1299).