

**MALWARE DETECTION FROM FILE IMAGES USING DEEP  
LEARNING ALGORITHM**

**Sathyabama Institute of Science and Technology  
(Deemed to be University)**

Submitted in partial fulfilment of the requirements for the award of  
Master of Engineering Degree  
in  
Computer Science and Engineering  
By

**RIDHI MAYURA SHIVANI R  
(Reg. No. 41310002)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119,  
TAMILNADU.**

**MAY 2023**



**SATHYABAMA**  
INSTITUTE OF SCIENCE AND  
TECHNOLOGY (DEEMED TO BE  
UNIVERSITY)

Accredited with Grade "A" by  
NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)



---

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Phase-2 Report is the bonafide work of **RIDHI MAYURA SHIVANI R (Reg no:41310002)** who out the project entitled "**Malware detection from file images using deep learning**" under your supervision from January 2023 to May 2023.

Internal Guide

Mrs. M. GOWRI M.E.,

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.,



Submitted for Viva voce Examination held on 25.4.2023

---

Internal Examiner

External Examiner

**DECLARATION**

I, **RIDHI MAYURA SHIVANI R (41310002)**, hereby declare that the Project Phase-2 Report entitled “**Malware detection file images using deep learning algorithm**” done by me under the guidance of **Mrs. M. Gowri M.E.**, is submitted in partial fulfillment of the requirements for the award of Master of Engineering degree in **Computer Science and Engineering**.

**DATE:25.4.2023**

**PLACE : Chennai**



**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.M.Gowri M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

Malware is one of the dangerous and most malicious threat to the digital systems. The increased number of users of internet, have increased the users exposure towards the malware. The traditional techniques such as anti-virus, anti-malware, firewalls etc., have not been so effective when it comes to the ever-evolving malware attacks. Malware always finds its own way to enter into the victims systems. In this project, an artificial intelligence based malicious file detection from image files has been proposed. We detect such kind of malware in the system (both the static and dynamic) by using the converted the SFC images, using scurve library in python. And these SFC images are used for feature extraction. The feature extraction is done using CNN algorithm and from these extracted features the required feature are selected and used for malware detection.

## TABLE OF CONTENTS

| CHAPTER NO | TITLE                                          | PAGE NO |
|------------|------------------------------------------------|---------|
|            | ABSTRACT                                       | V       |
|            | LIST OF FIGURES                                | Vii     |
| 1          | INTRODUCTION                                   | 1       |
|            | 1.1 OBJECTIVES                                 | 3       |
| 2          | LITERATURE SURVEY                              | 4       |
| 3          | AIM AND OBJECTIVE                              | 11      |
|            | 3.1 Aim                                        | 11      |
|            | 3.2 Objective                                  | 11      |
| 4          | DESCRIPTION OF PROJECT                         | 12      |
|            | 4.1 Existing system                            | 12      |
|            | 4.2 Proposed system                            | 13      |
|            | 4.3 Architecture diagram                       | 13      |
|            | 4.4 Modules                                    | 14      |
|            | 4.4.1 Data gathering and pre-processing        | 14      |
|            | 4.4.2 Feature extraction and Feature selection | 14      |
|            | 4.4.3 Malware detection                        | 15      |
|            | 4.4.4 Proposed algorithm                       | 16      |
| 5          | RESULTS                                        | 20      |
| 6          | CONCLUSION                                     | 24      |
|            | REFERENCES                                     | 25      |

## **LIST OF FIGURES**

| <b>CHAPTER NO</b> | <b>TITLE</b>                                     | <b>PAGE NO</b> |
|-------------------|--------------------------------------------------|----------------|
| 4                 | 4.3 Architecture diagram                         | 13             |
| 4                 | 4.4.1.1 Malware dataset                          | 14             |
| 4                 | 4.5.1 CNN Architecture                           | 19             |
| 5                 | 5.1 Malware features                             | 20             |
| 5                 | 5.2 Classification of malware and benign dataset | 20             |
| 5                 | 5.3 Pre-processing of dataset                    | 21             |
| 5                 | 5.4 Training and testing dataset                 | 21             |
| 5                 | 5.5 Loss and accuracy                            | 22             |
| 5                 | 5.6 Confusion matrix of model                    | 23             |

# CHAPTER 1

## INTRODUCTION

Malware word defines from Malicious Software. Malware is a malicious code that affects the user system or computer and intently harms the computer by an attacker. Malware is variant forms which are a virus, Trojan, backdoor, rootkits, ransomware, worm, botnet, spyware, adware, keyloggers, etc., and there is a wide range of their families are existing and massively growing on the internet daily.

Incidents of malware attacks are continually rising, with antivirus (AV) vendors, such as Kaspersky, reporting an average of 360,000 malware samples received daily (Kaspersky, 2020). Mal ware's relentless growth can be attributed to the financial rewards that can be gained from malware exploits such as crypto-currency coin miners, banker Trojans and ransomware. According to AVTest (2020), approximately 87% of malware processed in 2020 were not original programs, but rather variants of existing families. This presents a major problem to AV companies in terms of the pro cessing overheads required to analyse every sample. Automated tools can help alleviate the burden, but generally are configured for detection rather than classification. Analysis is further hampered by malware obfuscation. Obfuscation has legitimate uses in computing to secure applications from copying or to protect intellectual property, but is also widely adopted by malware developers to evade detection from AV scanners and slow down the analysis process. For the most part, static analysis is not robust against obfuscation, due to the upheaval of the binary structure of the malware program caused by these obfuscation methods. Dynamic approaches overcome the obfuscation limitations of static analysis, since the programs must be de-obfuscated and written to main memory prior to execution. However, dynamic analysis can produce a considerable number of false positives, due to similarities in behaviour to benign programs or other families of malware. Hybrid methods, where elements of both static and dynamic approaches are combined, have been presented to overcome these limitations.

According to the survey conducted by AV-Test Institute, it registers that everyday 350,000 new malicious code and potentially unwanted applications. Each malicious one is classified with respect to their behaviour and saved accordingly by this institute and gives the malware statistics in 2018 is 847.34m malicious code is found and recorded and registered. Some of the malware attacks in history are Melissa was a macro embedded with a word file. When the user opens it the macro will execute and resend the virus to the first 50 people in the user's address book. It was designed by David L. Smith in 1999. Likewise, several malware attacks in history namely, My Doom worm in 2004, Stuxnet in 2010, wannacry in 2017.

There is an endless competition between malware developers and analysts. Both research communities are working in parallel, one is developing the malware detection system and other is designing the malicious software to compromise the detection software for attacking the computer and networks resources.

Malware analysts analyze the known malware and aim to detect the malware to avoid the attack on the computer systems. Malware are detected using either signature-based or behaviour-based techniques. The signature based malware detection systems are fast and efficient but can be easily evaded by the obfuscated malware. On the other side, behaviour-based techniques are more resilient to the obfuscation technique. However, behaviour-based techniques are very time consuming. Not only the signature and behaviour-based malware detection techniques have been developed but also many hybrid techniques have been developed to inherit the advantages of both techniques. Hybrid techniques are designed to handle the issues of signature based and behaviour-based approaches.

The detection of zero-day malware is challenging because these malware exploits the new vulnerabilities which are not discovered yet.

Whenever new software is developed the crackers try to find the vulnerability in that and exploit that to compromise the security of the software. Therefore, there is a need to patch and develop the malware detection systems which can handle the severe attacks. Since the inception of malware attacks on the

computer system, the defending system has been developed. In this project it is proposed to classify and detect the kind of malware using deep learning algorithm CNN. The classification is done using the extracted features of image file and then the malware is detected.

malware is generally impervious to basic analysis methods and so analysts must use more invasive techniques to extract signatures for classification, which are inevitably not scalable due to their complexity.

### **1.1 Objective:**

The main objective of the proposed project is to detect Malware in a file, classify them into different families to defend computer, mobile and other device with accurate solution in less time.

## **CHAPTER 2**

### **LITERATURE SURVEY**

In this section, we review related visualisation-based techniques applied to malware analysis.

#### **Classification of malware detection**

Proposed by P HarshaLatha, R Mohanasundaram in 2020 used algorithms Naive Bayes, Random Forest, Support Vector Machine for the result Supervised Dimensionality Reduction in a large data set of malware samples.

#### **Network based analysis and classification of malware using behavioral artifacts ordering**

Proposed by Aziz Mohaisen , Omar Alrawi, Jeman Park, Joongheon Kim, DaeHun Nyang, Manar Mohaisen in 2019 did malware detection in this paper considered order-based behavioral features for classification of malware samples in its simplest form. Addressing the limitations outlined in Section 6.2 is the immediate future work. In particular, we would like to explore partial-order features for fingerprinting malware samples. Those features would address noised features (both intentional obfuscation by malware authors, and unintentional, due to mixed signals on the wire). Realizing the applications listed in Section 6.3 using the Chatter methodology is another future work that we would like to explore.

#### **Feature selection to classify a malware**

Proposed by Mouhammd Al-Kasassbeh, Safaa Mohammed, Mohammad Alauthman, and Ammar Almomani in 2020 , in this paper a presentation of seven key features was provided to aid in separating the programs that are clean or have malware. The identification of such features was done through the intuition that various parts of the PE files' features can correlate with one another less than with the class files, being clean or dirty. Such features are implemented as algorithms in machine learning to help classify the malware, resulting in such classification to be properly implemented in antivirus programs to help enhance the rate of detection.

### **Image based malware classification hybrid framework based on space filling curves**

Proposed by Stephen O'Shaughnessy , Stephen Sheridan in 2022, in this paper they presented a novel hybrid framework for malware classification, based on computer vision and machine learning, designed to improve the current limitations experienced in static and dynamic malware variant classification methods. The framework is designed to be less invasive than traditional analysis methods in that there is no reverse engineering required, nor does it suffer from the obfuscation limitations of static analysis. The framework incorporates classification of discriminant features extracted from malware binaries in image format, mapped through space-filling curves. Previous research has shown that space-filling curves provide sufficient features to be used for image-based malware classification. Through experimental analysis, using an up-to-date malware sample dataset, it was demonstrated that the hybrid framework outperformed the static and dynamic analyses standalone methods. The methods produced by this research could be used to augment the performance of automated malware analysis tools by increasing the classification rates of variants and thereby reducing the possibility of ambiguity in malware labelling.

### **A distance based method for building an encrypted malware traffic identification framework**

Proposed by JIAYONG LIU, ZHIYI TIAN , RONGFENG ZHENG , LIANG LIU in 2019, the proposed model can measure similarity between malwares and form an adaptive identification framework, according to the distribution of traffic feature dataset. The experiment showed that the proposed method was able to identify multiple encrypted malware traffic with higher accuracy and less identification time.

Moreover, the proposed method can deal with unknown traffic.

### **Byte level malware classification**

Proposed by Baoguo Yuana , Junfeng Wang b , Dong Liuc , Wen Guoc , Peng Wua , Xuhua Bao in 2020, they did static reverse analysis and dynamic

analysis respectively have their own limitations, traditional machine learning algorithms are generally difficult to process massive unknown malware samples. This paper proposed a byte-level malware classification method called MDMC which converted malware binaries into markov images and classified malware markov images with deep learning. Only the binaries of malware were used without the reverse analysis and dynamic analysis. MDMC could be applicable to various systems such as windows and android. Compared with similar methods such as GDMC, MDMC could significantly improve the accuracy of malware classification.

### **Malware detection using image visualisation**

Proposed by Abdullah Sheneamer, Essa Alhazmi, James Henrydoss in 2022, the proposed that In this work, we implement the computer visualizationbased technique to build ML-based malware detection. We employ a GIST-based approach to extract malware image features from both the gray-scale and color images of malware binary samples. To study the prediction performance, we empirically analyze various machine learning algorithms. Our experimental study includes the following learning algorithms: linear regression, random forest, k-NN, CART, SVM, and MLP, and a CNN-based deep learning model and selects a candidate learning classifier that can yield better prediction performances. We evaluate our approach using the following malware datasets SARVAM, PhD-thesis, and CIC. In comparison to traditional malware detectors, the visualization-based approach provides a significant performance enhancement. Our study observes that the CNN-based deep-learning model yields significantly better performance when tested against the various malware datasets listed above. Malware authors constantly innovate new methods for implementing sophisticated attacks that make the malware detection as an active research area. Our approach provides a path forward to implement an innovative malware detector. Nevertheless, this research work needs to be verified against various new malware datasets to be more effective. In the future, we would like to extend our research to benchmark datasets and as well conduct a large-scale evaluation.

## **Malware detection in computer**

Proposed by Mohammed et al, in 2020, identified malware as a computer security issue used to avoid the normal detection methods which rely on signature matching. Those that contain identical malware variants and patterns are used for the machine techniques in order to identify others. Comparative study of various selection methods used by the KNN algorithm to classify the malware based on the n-grams analysis, resulting in the use of “principal component analysis (PCA)” feature selection in addition to the “support vector machines (SVM)” classification to provide the best classification with the minimum features. In this study, the dataset used includes the 1156 malware files in addition to 984 benign files of different formats; the obtained accuracy in this paper is equal to 93%.

## **Malware detection using ssn algorithm**

Liu et al, suggested a learning malware system analyzed through machines to contain three parts. One is the data processing and malware detection. The other is decision-making. The first deals with Opcode n-gram and the gray-scale images in addition to importing functions to extract the malware features. The latter tends to implement the features to classify and identify the files which are suspicious of malware. The dataset in this paper includes “20000 malware instances” that had been collected by Anubis, Kingsoft, as well as ESET NOD32. Lastly, the detection part would put to use “shared nearest neighbor (SNN) algorithm” to identify the new malware family, resulting in this system classifying with a percentage of 98.9% accuracy and 86.7% identifying the specific malware. **Classification of malware using Genetic algorithm**

Lin et al, using n-grams features from dynamic and static features. Also, the authors used approximately 790,000 n-grams and proposed a genetic algorithm to classify malware. They implemented feature reduction and selection, and also they obtained an accuracy equal to 90% using 10 features as well as 96% using 100 features. In this paper, the dataset was 389 benign samples in addition to 3899 malware. Feature reduction and selection had been performed on the determined number of n-grams; it is not strictly

associated with the number of features, which is needed in order to read from the behaviour of dynamic and static from the files.

### **Malware detection using Kernel method algorithm**

Ismail et al, extracted that n-gram from packet payloads finding malware at the infrastructure level of a network to later suggest that there is a similar pattern used to detect solely based on the idea that the malware would carry certain features of the old one. The dataset included 2507 training flows as well as the test dataset of 3470 flows in addition to 27,491 training flows, and the used algorithm was Kernel method algorithm. Then later incorporated this domain knowledge which he derived from the SNORT signatures using a Naive Bayes classifier, by using this information they came to find that their experiment work which used only a few features of only 90,000 features put to utilize, therefore to reduce the processing time from 53 h to only 3 by solely optimizing the number of those features.

### **Detection of malware using KNN and SVM in addition to the decision tree**

Kong et al, provided a structure using malware which was automated and solely based on the informational structure of that specific malware. The dataset included 526,179 malwares, and the used algorithms are KNN and SVM in addition to the decision tree. Upon extracting the features using the cell graphs of each one, the authors came to conclude that there were similarities in the two programs of malware. All this is by implementing the distance metric learning to cluster them and sort them depending on family and maintaining the different ones at a margin; then they finally assembled those using an ensemble in order to classify the malware and include them in the family which they belong to (Table 36.2 summarize of malware detection methods).

### **Visualisation of binary files**

The first study on the visualisation of binary files as images was introduced in the work of Conti et al. (2008), which presented a method of reverse engineering binary files into visual images, dubbed byte plots, to enhance the capabilities of text-based hex editors. Conti et al. used the byte plot method

to create a visual taxonomy of binary file fragments to aid in the identification of common file formats. In (Conti et al., 2010), the authors extended their previous work using a K-nearest neighbours classifier to classify file types by their visual features, trained on a dataset of 14k samples comprising 14 file types, including random data, encryption, compression, machine code, text and bitmap images. The best results reported were from unencoded and base64 encoded files, with 100% accuracy, with plain text files scoring 98.7%. While the initial work presented by Conti et al. was not in the malware domain, it demonstrated that the internal static structure of binary files could be represented as 2-dimensional images, which paved the way for further research efforts and tools that utilised byte plots for malware classification.

### **Byteplot malware visualisation**

Nataraj et al. (2011a) first applied byteplots to represent malware binaries for classification. Their method visualized static malware samples as gray-scale images, with the observation that for many malware families, the images belonging to the same family appear very similar in layout and textures. Malware feature vectors were derived using GIST, a global feature descriptor first presented by Oliva and Torralba (2001), that breaks images into sub-band blocks and computes features based on filters tuned to varying scales and orientations. The authors compiled a dataset, dubbed Maling, comprising 9548 samples from 25 malware families. Using a K-nearest neighbours (KNN) classifier, Nataraj et al. reported a classification accuracy of 98%. Nataraj et al. (2011b) followed on from their original research, presenting a comparison of the byteplot method with dynamic analysis. They used a combination of API hooking and pre-infection and post-infection system snapshots to build dynamic feature vectors. Results showed that overall, image classification was comparable in terms of classification accuracy with dynamic analysis methods but outperformed the latter in terms of reduced time complexity and robustness against obfuscated malware samples. While Nataraj et al. report the byteplot method as resilient to obfuscation, their solution to identifying packed malware was to treat it as a separate class. This approach does not solve the packed malware problem as it is identifying the packed and unpacked samples from

the same families as two different species. Furthermore, the authors do not consider the effect of encrypted malware samples on their GIST-based classification method. A number of subsequent research efforts applied the byteplot format to malware classification. Luo and Lo (2017) extracted visual features for classification using Tensorflow and Local Binary Patterns (LBP) feature descriptors on the Maling dataset. Results showed that LBP and Tensorflow convolutional neural network produced a classification accuracy of 93.17% as opposed to 82.83% using the method proposed by Nataraj et al. (2011a). Yajamanam et al. (2018) evaluated the GIST-based byteplot method using a subset of features of the original method. Further testing was carried out using deep learning with Tensorflow. Results were comparable with the original research of Nataraj et al. (2011a), while using a smaller feature set of 60 features. Le et al. (2018) used convolutional neural networks on binary files from the Microsoft Malware Classification Challenge dataset (Ronen et al., 2018) converted to byteplot images. Results on the data reported 98.8% accuracy on validation data with a fast processing time-frame of approximately 20ms per sample. Fu et al. (2018) extracted features from both grayscale and colour byteplots using local gray-level co-occurrence matrices and global colour moments, which were then passed to Random Forest, K-Nearest Neighbour and Support Vector Machine classifiers. The experiments were carried out on a dataset comprised of 7087 malware samples from 15 distinct malware families showed the combined feature sets performed better than either local or global features alone, scoring 97% precision, recall, accuracy and f-measure with the Random Forest classifier.

## **CHAPTER 3**

### **AIM AND SCOPE OF PRESENT INVESTIGATION**

#### **3.1 AIM**

The main objective of the proposed project is to detect Malware in a file, classify them into different families to defend computer, mobile and other device with accurate solution in less time

#### **3.2 SCOPE**

In this paper, the emphasis is put on the Deep Learning-based malware detection techniques to develop an intelligent malware detection system for the executable files. This paper discusses the proposed work for detecting the executable files and provides insight into ongoing research in malware detection using the various features and approaches.

## CHAPTER 4

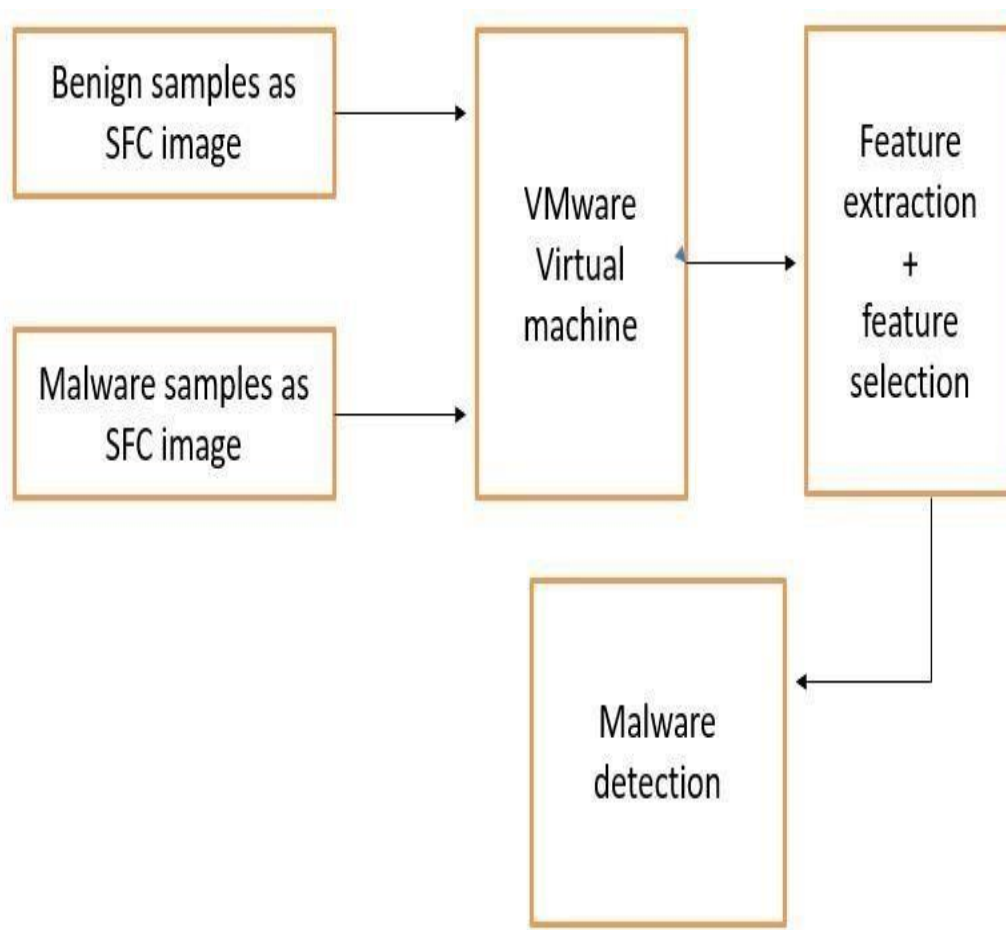
### DESCRIPTIONS OF PROPOSED SYSTEMS

**4.1 EXISTING SYSTEM** presented a novel hybrid framework for malware classification, based on computer vision and machine learning, designed to improve the current limitations experienced in static and dynamic malware variant classification methods. The framework is designed to be less invasive than traditional analysis methods in that there is no reverse engineering required, nor does it suffer from the obfuscation limitations of static analysis. The framework incorporates classification of discriminant features extracted from malware binaries in image format, mapped through space-filling curves. Previous research has shown that space-filling curves provide sufficient features to be used for image-based malware classification. Through experimental analysis, using an up-to-date malware sample dataset, it was demonstrated that the hybrid framework outperformed the static and dynamic analyses standalone methods. The methods produced by this research could be used to augment the performance of automated malware analysis tools by increasing the classification rates of variants and thereby reducing the possibility of ambiguity in malware labelling. Our current research is focused on implementing image segmentation for classification at a more granular level. Segmentation can be used to partition the gray images into distinct regions containing groups of pixels with similar textures, in this case regions of similar binary. Regions of interest in an executable binary could be extracted in this way, disregarding irrelevant data such as padding regions of binary zeros) or obfuscated sections, where the entropy would otherwise adversely affect the classification algorithms. Additionally intended to investigate the efficacy of other SFCs, such as the H-curve (Niedermeier et al., 2002), applied to the malware domain. Finally, deep learning models, such as convolutional neural networks, could be utilised to improve performance. Such models do not require feature extraction and so would simplify the processing phases.

## 4.2 PROPOSED SYSTEM

In this work, an effective approach is suggested for classification and detection of malware using deep learning algorithm CNN(Convolutional Neural Network). The framework incorporates novel static and dynamic malware analysis methods. The 2-D color images that is mapped through space filling curve are taken as input and fed into the convolutional neural network layer .This method is provides dynamic as well as statistical analysis.This method processes non-obfuscated samples statically and obfuscated samples dynamically. Non-obfuscated samples do not require any behavioural dynamic analysis because sufficient features can be captured from static binary. Dynamic analysis captures the behavioural data of the dynamic data. Computer vision techniques are then used to extract discriminant features from the images for malware detection.The features which are extracted are used for malware detection process.

## 4.3 SYSTEM ARCHITECTURE



## 4.4 MODULES

### 4.4.1 GATHERING OF DATA AND PREPROCESSING

There are a few malware datasets available for academic research. One of these datasets is Maling. The dataset contains 9,342 malware images, classified into 25 malware families. The widths and lengths of the malware images vary. The images have been created from various malware families such as Dialer, Backdoor, Worm, WormAutoIT, Trojan, Trojan-Downloader, Rouge and PWS. All malware images are PE files that were first converted to an 8-bit vector binary, and then to images. The malware image sizes were modified, so that they can be input to a CNN model. The data collected in the database are pre-processed and selected only the families with more than 500 example.



Fig 4.4.1.1 Malware dataset

### 4.4.2 FEATURE EXTRACTION AND FEATURE SELECTION

Three feature descriptor algorithms were evaluated, namely Local Binary Patterns (LBP), Gabor filters and Histogram of Oriented Gradients (HOG) . While the HOG descriptor is primarily used for object detection in images, it

has produced favourable results in texture analysis . This section provides an overview of how each descriptor was applied in the context of this work.

In the case of each descriptor, a dataset of feature vectors was generated for every image, along with a dataset of corresponding family class labels. LBPs compute a local representation of texture by comparing and thresholding each pixel with its surrounding neighbourhood of pixels. The original LBP operator forms labels for the image pixels by thresholding the  $3 \times 3$  neighbourhood of each pixel with the centre value and considering the result as a binary number. The histogram of these  $2^8 = 256$  different labels are used as a texture descriptor, which can represent the grayscale range of 0 – 255.

In our framework, we used the circular pixel neighbourhood extension to the original LBP operator. The extension operator allows a greater number of neighbourhood points to be considered in the calculation, and thus can capture textures at varying scales of granularity. The main parameters to consider in the LBP descriptor are the radius of the circular neighbourhood and the number of data points on the circle's circumference. Values at non-integer pixel coordinates are bi-linearly interpolated which allows any radius and number of pixels in the neighbourhood. Once the feature are extracted from image the required features are selected

#### **4.4.3 MALWARE DETECTION**

By using the selected features from the above module , the binary file is verified whether it is a malware file or not. If it is malware then it can be further processed and can be identified whether the image is malware or not.

The objectives of malware detection methods are to detect and defend the malicious programs that can harm the computer system or network assets. The Input is represented in many ways to detect and classify the malware samples into an adequate family. The relevant information or knowledge about the malicious file is required which represents the actual behaviour of malware file. For that various mal ware samples are analyzed with static, dynamic or hybrid techniques. After that, the extracted information is represented into the proper format which is further utilized to train the detection system. Ucci et al.

(2019) [73] explained various different features which have been used by various researcher to develop the malware detection system such as Byte sequences, Opcodes, Strings, APIs calls, PE Header information, Windows Registry, File system accesses, AV/Sandbox submissions, CPU Registers, Length of functions, Network Activities and Generated Exceptions. Fundamentally, the malware files are analyzed; features are extracted and represented into an intermediate form before inputting to malware detection. This intermediate form plays a vital role in detection. If the extracted information is appropriate to detect the malware only then, the false-positive rate can be decreased. The various proposed malware detection techniques are classified into three main categories which are explained as follows In this approach, malware are detected based on a particular signature or pattern of the files. It is a conventional method which is very fast to detect known malicious files as compared to other techniques. Mostly, antivirus software are implemented using signature-based techniques. One simple way to create the signature of malware files using a hash algorithm like MD5, SHA1 etc. Signatures of malware files are generated and stored in the database of the detection system for verifying the signature of an unknown file. If the signature of the file is matched then, it is declared as malicious file otherwise benign file. The problem of this technique is that the signature of the file is changed even if a single byte of the file is changed. Thereby for each modified, malware and new malware, a new signature has to be generated. Only then, a malware detector can detect that malware. Supplementary, new signature-based malware detection systems were proposed which uses the other patterns such as program instructions, control flow graph, and mnemonic sequence. Fig. 4 shows the schematic architecture of signature-based techniques. As shown in Fig. 4 malware samples are analyzed using various tools IDA Pro, Lida, Peview, PeStudio and features are extracted. Extracted features are used to train the malware detector either using traditional or machine learning algorithms. This figure represents the all the possible ways to develop the signature-based malware classifiers using static features. it covers the various tools, static features and approaches to design malware detectors. In this

section, all the presented signature-based techniques have been proposed using this architecture.

#### **4.5 PROPOSED ALGORITHM**

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called **feedforward**, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that,

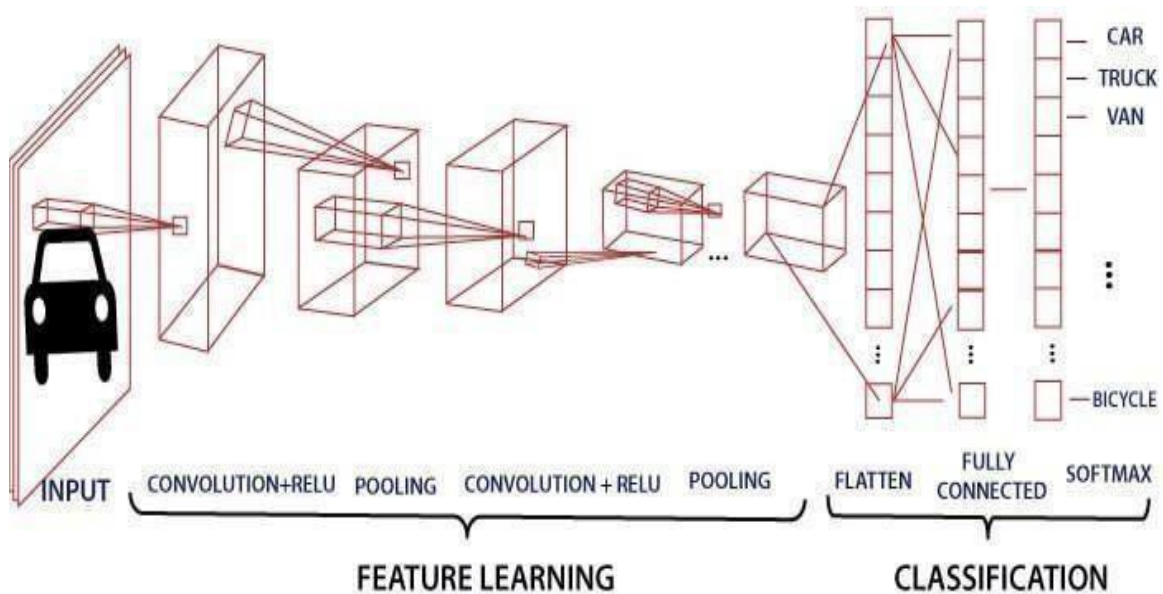
we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

The proposed technique is primarily separated into five sections: building the CNN model, computing confidence using the CNN model, choosing ground control points, fine-tuning matching costs, and computing final disparity maps. The data is fed into the model and output from each layer is obtained. In a CNN, the input image is passed through multiple convolutional layers, where each layer applies a set of filters to the input. These filters are learned during training and are used to identify patterns and features in the image. The output of each convolutional layer is then passed through a pooling layer, which reduces the dimensionality of the feature maps. Finally, the output of the last pooling layer is flattened and passed through one or more fully connected layers, which make the final prediction. CNNs have shown to be highly effective in image classification tasks and are widely used in various machine learning projects. The algorithm is based on the anatomy of the human visual system and is intended to learn and extract characteristics from photos automatically. The convolution, pooling, and activation processes carried out by the CNN algorithm are distributed over many layers of linked neurons. These layers can recognise intricate patterns in photos by extracting characteristics at various degrees of abstraction. CNN takes an image as input, which is classified and process under a certain cate gory such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as  $h * w * d$ , where  $h$ = height  $w$ = width and  $d$ = dimension. For example, 24 An RGB image is  $6 * 6 * 3$  array of the matrix, and the grayscale image is  $4 * 4 * 1$  array of the matrix. In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also

known as kernels). After that, we will apply the Softmax function to classify an object with probabilistic values 0 and 1.



**Fig 4.5.1 cnn architecture**

CNN will do forward and backward propagation to do predictions and gradient computations. Following is the forward propagation

$$x_{ij}^l = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{l-1}$$

## 4.6 SYSTEM REQUIREMENTS

### 4.6.1 software requirements

Python

Vmware virtual box

### 4.6.2 hardware requirements

Keyboard

Mouse

8 Gb ram

## CHAPTER 5

### RESULTS

| md5                | Machine | SizeOfOptionalHeader | Characteristics | MajorLinkerVersion | MinorLinkerVersion | SizeOfCode | SizeOfInitializedData | SizeOfUninitialize |
|--------------------|---------|----------------------|-----------------|--------------------|--------------------|------------|-----------------------|--------------------|
| 9b4707448e442b53a8 | 332     | 224                  | 258             | 9                  | 0                  | 361984     | 115712                |                    |
| 289acd5641e3a7ea8b | 332     | 224                  | 3330            | 9                  | 0                  | 130560     | 10960                 |                    |
| 13c0db88a70bdc0360 | 332     | 224                  | 3330            | 9                  | 0                  | 517120     | 821568                |                    |
| 70074b07805f0c9e12 | 332     | 224                  | 258             | 9                  | 0                  | 585728     | 389152                |                    |
| 8650ce8999b543a731 | 332     | 224                  | 258             | 9                  | 0                  | 294812     | 247296                |                    |

Fig 5.1 Malware features

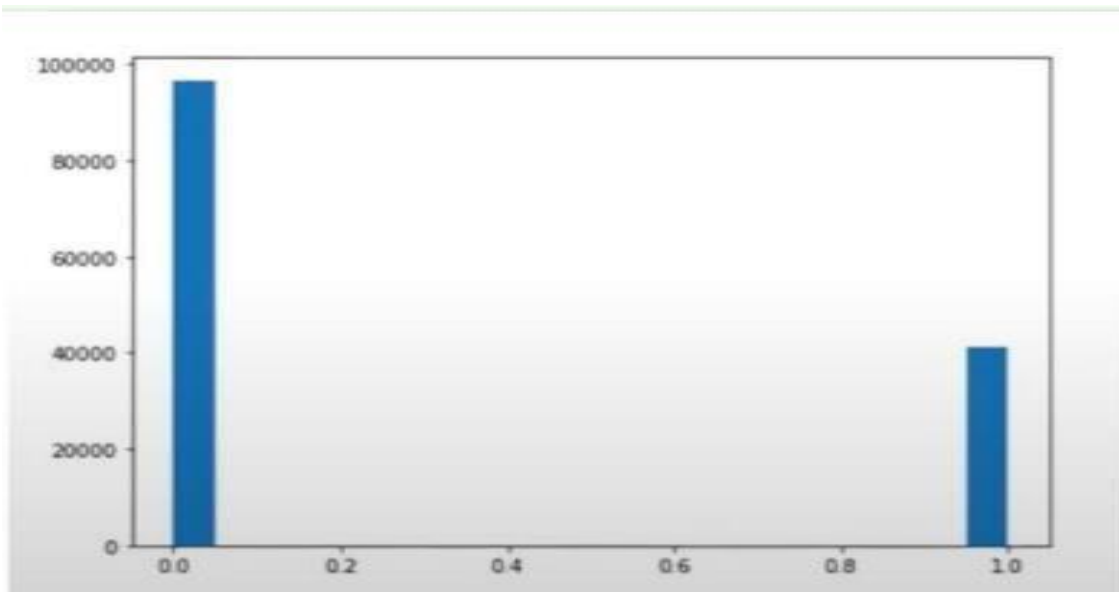


Fig 5.2 classification of Malware and benign dataset



Fig 5.3 Pre- processing of data

Model: "sequential\_7"

| Layer (type)            | Output Shape | Param # |
|-------------------------|--------------|---------|
| dense_28 (Dense)        | (None, 16)   | 880     |
| dense_29 (Dense)        | (None, 8)    | 136     |
| dense_30 (Dense)        | (None, 4)    | 36      |
| dense_31 (Dense)        | (None, 1)    | 5       |
| Total params: 1,057     |              |         |
| Trainable params: 1,057 |              |         |
| Non-trainable params: 0 |              |         |

Fig 5.4 Training and testing dataset

```
Epoch 1/5
3452/3452 [=====] - 3s 727us/step - loss: 6907936.0000 - accuracy: 0.9472
Epoch 2/5
3452/3452 [=====] - 2s 663us/step - loss: 6835772.0000 - accuracy: 0.9533
Epoch 3/5
3452/3452 [=====] - 2s 675us/step - loss: 3920674.5000 - accuracy: 0.9470
Epoch 4/5
3452/3452 [=====] - 2s 677us/step - loss: 4201450.5000 - accuracy: 0.9471
Epoch 5/5
3452/3452 [=====] - 2s 675us/step - loss: 2288503.0000 - accuracy: 0.9462

<tensorflow.python.keras.callbacks.History at 0x20c96bbe7c8>
```

Fig 5.5 Loss and accuracy

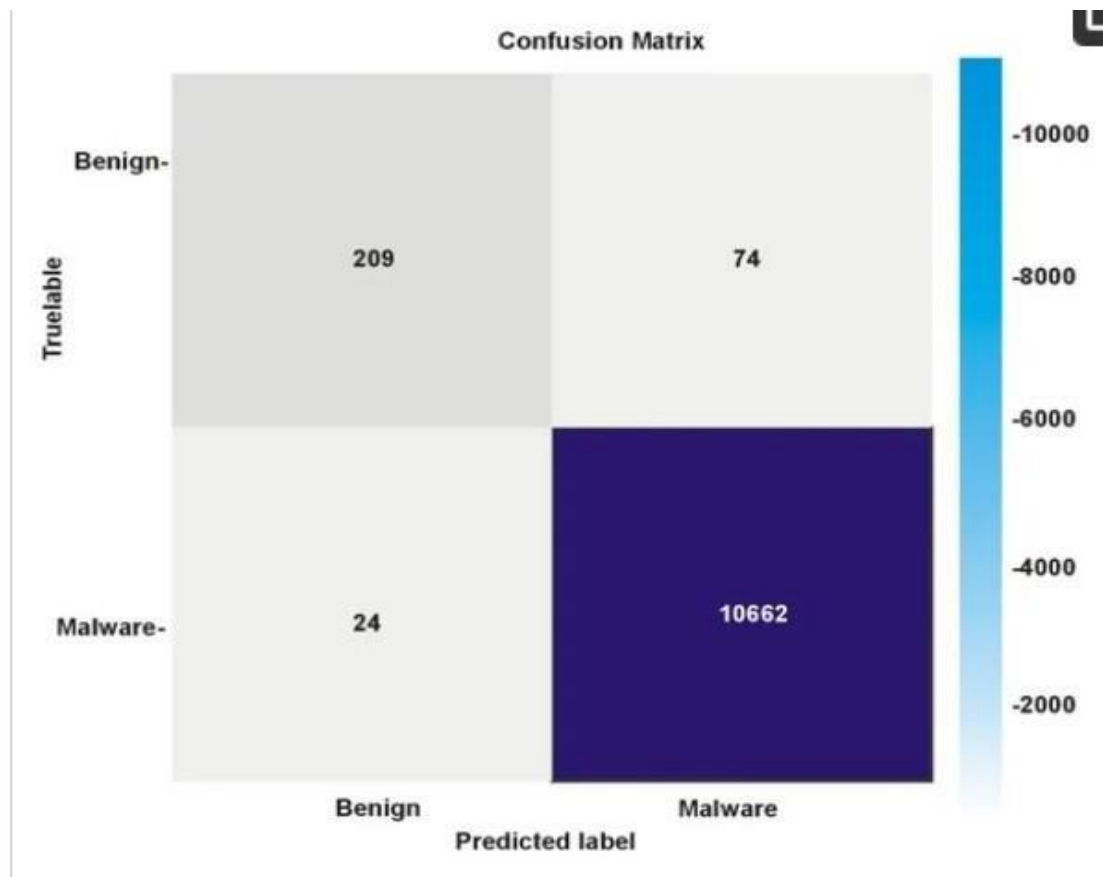


Fig 5.6 confusion matrix

## **CHAPTER 6**

### **CONCLUSION**

In this project, we presented a novel hybrid framework for malware detection , based on cnn a deep learning algorithm, designed to improve the current limitations experienced in static and dynamic malware variant classification methods. The framework is designed to be less invasive than traditional analysis methods in that there is no reverse engineering required, nor does it suffer from the obfuscation limitations of static analysis. The framework incorporates extraction of discriminant features from malware binaries in image format, mapped through space-filling curves. And malware detection by using the extracted features.

## REFERENCES

- [1] Anderson, B., Storlie, C., Lane, T., 2012. Improving malware classification: Bridging the static/dynamic gap. Association for Computing Machinery, New York, NY, USA, pp. 3–14. doi:10.1145/2381896.2381900.
- [2] AV-Test, 2020. Malware statistics & trends report. <https://www.av-test.org/en/statistics/malware/>. Baptista, I., 2018. Binary visualisation for malware detection. *The Plymouth Student Scientist* 11, 223–237.
- [3] Conti, G., Bratus, S., Shubina, A., Lichtenberg, A., Sangster, B., Supan, M., 2010. A visual study of primitive binary fragment types. doi:10.1016/j.diin.2010.05.002.
- [4] Conti, G., Dean, E., Sinda, M., Sangster, B., 2008. Visual reverse engineering of binary and data files. In: Goodall, J.R., Conti, G., Ma, K.-
- [5] L. (Eds.), *Visualization for Computer Security*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–17.
- [6] Cortesi, A., 2012. scurve, a space-filling curve library. <https://github.com/cortesi/scurve>. Demir, H., 2018. Classification of texture images based on the histogram of oriented gradients using support vector machines. *Istanbul University - Journal of Electrical and Electronics Engineering* 18, 90–94.
- [7] J. Shen, J. Xia, X. Zhang, and W. Jia, “Sliding block-based hybrid feature subset selection in network traffic,” *IEEE Access*, vol. 5, pp. 18179–18186, 2017.
- [8] F. Iglesias and T. Zseby, “Analysis of network traffic features for anomaly detection,” *Mach. Learn.*, vol. 101, nos. 1–3, pp. 59–84, 2015.
- [9] M. Shafiq, X. Yu, A. K. Bashir, H. N. Chaudhry, and D. Wang, “A machine learning approach for feature selection traffic classification using security analysis,” *J. Supercomput.*, vol. 74, no. 10, pp. 4867–4892, 2018. doi: 10.1007/s11227-018-2263-3
- [10] S.M. Muzammal, M.A. Shah, S.J. Zhang, H.J. Yang, Conceivable security risks and authentication techniques for smart devices: A comparative evaluation of security practices, *Int. J. Autom. Comput.* 13 (4) (2016) 350–363, <http://dx.doi.org/10.1007/s11633-016-1011-5>. 26

- [11] J. Singh, J. Singh, Challenges of malware analysis : Obfuscation techniques, *Int. J. Inf. Secur. Sci.* 7 (3) (2018).
- [12] Y. Gao, Z. Lu, Y. Luo, Survey on malware anti-analysis, in: 5th International Conference on Intelligent Control and Information Processing, ICICIP 2014 - Proceedings, 2015, pp. 270–275, <http://dx.doi.org/10.1109/ICICIP.2014.7010353>.
- [13] S. Alam, R. Horspool, I. Traore, I. Sogukpinar, A framework for metamorphic malware analysis and real-time detection, *Comput. Secur.* 48 (2015) 212–233, <http://dx.doi.org/10.1016/j.cose.2014.10.011>, arXiv:arXiv:1011.1669v3, <http://linkinghub.elsevier.com/retrieve/pii/S0167404814001576>.
- [14] J. Singh, J. Singh, Ransomware: an illustration of malicious cryptography (2) (2019), 1608–1611, <http://dx.doi.org/10.35940/ijrte.B2327.078219>.
- [15] X. Hu, Large-scale malware analysis, detection, and signature generation (ProQuest Dissertations and Theses), 2011, p. 190, <http://search.proquest.com/docview/918832186?accountid=44888>.
- [16] P. Coogan, Spyeye bot versus zeus bot, 2010, <http://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot>.
- [17] N. Eltayieb, R. Elhabob, A. Hassan, F. Li, A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud, *J. Syst. Archit.* 102 (August 2019) (2020) <http://dx.doi.org/10.1016/j.sysarc.2019.101653>.
- [18] D. Jang, Y. Jeong, S. Lee, M. Park, K. Kwak, D. Kim, B.B. Kang, Rethinking anti-emulation techniques for large-scale software deployment, *Comput. Secur.* (2019) <http://dx.doi.org/10.1016/j.cose.2019.02.005>.
- [19] S. MahdaviFar, A.A. Ghorbani, Application of deep learning to cybersecurity: A survey, *Neurocomputing* 347 (2019) 149–176, <http://dx.doi.org/10.1016/j.neucom.2019.02.056>, <http://www.sciencedirect.com/science/article/pii/S0925231219302954>.
- [20] W. Zhang, H. Wang, H. He, P. Liu, DAMBA: Detecting android malware by ORGB analysis, *IEEE Trans. Reliab.* 69 (1) (2020) 55–69, <http://dx.doi.org/10.1109/TR.2019.2924677>. 27

- [21] L. Liu, B.-s. Wang, B. Yu, Q.-x. Zhong, Automatic malware classification and new malware detection using machine learning, *Front. Inf. Technol. Electron. Eng.* 18 (9) (2016) 1–12, <http://dx.doi.org/10.1631/FITEE.1601325>.
- [22] R. Kaur, M. Singh, Hybrid real-time zero-day malware analysis and reporting system, *Int. J. Inf. Technol. Comput. Sci.* 8 (4) (2016) 63–73, <http://dx.doi.org/10.5815/ijitcs.2016.04.08>, <http://www.mecs-press.org/ijitcs/ijitcs-v8-n4/v8n4-8.html>.
- [23] J. Milosevic, M. Malek, A. Ferrante, Time, accuracy and power consumption tradeoff in mobile malware detection systems, *Computers & Security* 82 (2019) 314–328, <http://dx.doi.org/https://doi.org/10.1016/j.cose.2019.01.001>, <http://www.sciencedirect.com/science/article/pii/S0167404818307880>.
- [24] H. Studiawan, F. Sohel, C. Payne, A survey on forensic investigation of operating system logs, *Digital Investigation* 29 (2019) 1–20, <http://dx.doi.org/https://doi.org/10.1016/j.diin.2019.02.005>, <http://www.sciencedirect.com/science/article/pii/S1742287618303980>.
- [25] B. Ndibanje, K.H. Kim, Y.J. Kang, H.H. Kim, T.Y. Kim, H.J. Lee, Applied sciences cross-method-based analysis and classification of malicious behavior by API calls extraction, 2019, <http://dx.doi.org/10.3390/app9020239>