

MALWARE DETECTION USING SPACE FILLING CURVES AND BINARY TO IMAGE CONVERSION

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

SANDHYA RANI ILLURI

(Reg. No. 41310004)



**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

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Phase-2 Report is the bonafide work of **SANDHYA RANI ILLURI (Reg no:41310004)** who out the project entitled "**Malware detection using space filling curves and binary to image conversion**" 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

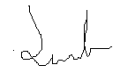
External Examiner

DECLARATION

I, **SANDHYA RANI ILLURI (41310004)**, hereby declare that the Project Phase-2 Report entitled “**Malware detection using space filling curves and binary to image conversion**” done by me under the guidance of **MS.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 **MS.M.Gowri M.E. , for her** valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 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 has been proposed. We detect such kind of malware in the system (both the static and dynamic) by converting the binary files into 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 these extracted features are used for malware classification and detection.

TABLE OF CONTENTS

Chapter no	TITLE	Page no
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	OBJECTIVES	4
3	LITERATURE SURVEY	5
4	SYSTEM REQUIREMENTS	13
	4.1 Hardware requirements	13
	4.2 Software requirements	13
5	DESCRIPTION OF PROPOSED SYSTEM	14
	5.1 Existing system	14
	5.2 Proposed system	15
	5.2.1 Space filling curves	15
	5.3 Architecture diagram	18
	5.4 Modules	19
	5.4.1 Data gathering and preprocessing	19
	5.4.2 Mapping binary files	22
	5.4.3 Malware classification	25
	5.4.4 Malware detection	31
6	RESULTS AND CONCLUSIONS	32
7	REFERENCES	35

LIST OF FIGURES

Chapter no	Title	Page no
5	5.1 Mapping data through SFC curves	17
5	5.2 Architecture diagram	18
5	5.3 Converted image	24
5	5.4 Different families of malware	29
5	5.5 CNN soft max function	30
5	5.6 Classified malware families	31
6	6.1 Bar graph of malware families	33
6	6.2 Test and train of malware	33
6	6.3 Confusion matrix	34

CHAPTER 1

INTRODUCTION

Malware is nothing but a malicious software in the form of a program or a file. These malwares are mostly intentionally sent into the systems that causes harm to the computer. These kind of malwares can be used for stealing personal information like user names and passwords. Studies have stated that 41% malware used by the hackers or attackers are unknown and only 59% of the malware is known until now. And 23% of publicly available malware is used by a range of attackers with varying skill levels. Some of them are used to corrupt the system files or delete the files or replace the data in the files. There are different kind of malwares , some run in the background and collects the user's login sessions and confidential information , in this case the user is unaware of the malware inside the system . And some malware makes the systems run slow , consumes all the CPU space etc.. this kind of malware can be detected by user.

Incidents of malware attacks are continually rising, with anti-virus (AV) vendors, such as Kaspersky, reporting an average of 360,000 malware samples received daily (Kaspersky, 2020). Malware'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 AV Test (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 processing 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

In this project we are going to detect all the kinds of malwares that are acting on the system by converting the binaries into SFC images using the scurve library . The 2-dimensional image of the binaries has been formed to reduce the lose of originality. This framework combines the both static and dynamic features which are gathered from the malware samples and convert them into 2-dimensional images . The 2-dimensional images are mapped through space-filling curves (SFC) traversals . Computer vision techniques are used to extract different features from the images for detection.

In this framework, the static and dynamic features that are collected from all the malware and benign samples are converted into 2-dimensional images which are mapped through space filling curve traversals . The space filling which is used over here is the hilbert curve. In general , the space filling curve traversals contain the whole n-dimensional unit hypercube. In this project , we consider only the 2-dimensional units because our output is the 2-dimensional image. In this case, the 2-dimensional image which is converted represents the image which is of size $n \times n$ pixels. The space filling curve passes through every pixels point in the general spatial region in a way that every spatial point of the given data is preserved exactly i.e the points that lie closer to each other in the binary file format (1-dimensional space) will remain closer to each other even after they got converted into the 2-dimensional image format. In this project, we present a novel malware classification hybrid framework based on computer vision techniques. Computer vision has been previously demonstrated as a viable alternative to the more typical methods for malware classification, such as static frequency and sequence-based approaches or dynamic API call analysis. Image-based approaches to malware classification involve the conversion of malware binaries to 2- dimensional images, such that characteristic properties or features. Computer vision techniques are then used to extract discriminant features from the images for classification. Malware classification through computer vision can be considered less invasive than the

analysis approaches mentioned previously, as no reverse-engineering or domain-specific knowledge is required, so it is less complex and thus can improve scalability for the processing of large malware datasets.

Our framework combines static and dynamic features gathered from malware samples converted to 2-dimensional images, mapped through space-filling curve (SFC) traversals. In mathematical analysis, SFCs are curves whose ranges contain the entire 2-dimensional unit square or more generally, an n -dimensional unit hypercube. However, for the purposes of this research, only the 2-dimensional space is considered, since the outputs are 2-dimensional images. In this case, the 2-dimensional unit square represents an image of $n \times n$ pixels. SFCs pass through every pixel point of a regular spatial region, such that the spatial locality of the data is preserved, i.e., closely located points in the binary file space will also be closely located when mapped to an SFC image. This is of significant importance, as the data structures within the original malware binaries can be retained in the resulting SFC images and thus used to represent the malware for classification purposes. Our hybrid method processes non-obfuscated samples statistically and obfuscated samples dynamically. Non-obfuscated samples do not require behavioural dynamic analysis because sufficient features can be captured from the static binary. Dynamic analysis can be considered more invasive since malware samples must be executed in a controlled environment to capture their behavioural data and then written to physical memory on the host. Additionally, the time complexity for a fully dynamic analysis approach would make it infeasible for large scale datasets.

CHAPTER 2

OBJECTIVE

The main objective of the project is to detect the different families of malware acting on a computer system. Some of these families include SHORTBENCH , TRICKBOT , QAKBOT , EMPIRE , BEACON ,ADIALER.C , AGENT.FYI , ALLAPLE.L ,C2LOP.P , DIALPLATFORM.B , FAKEREAN , LOLYDA.AA1 , LOLYDA.AT , MALEX.GEN!J, OBFUSCATOR.AD , SKINTRIM.N , SWIZZOR.GEN!E , WINTRIM.BX , YUNER.A and some others . Here we are using 24 different families of malware to train the system to detect the malware. This is done by image based detection approach and some computer vision techniques. The 2-dimensional images are created using space filling curves like z-code, Hilbert , grey code curves . The resulting output is represented in the form of 2-dimensional images . Different SFC images are made using the s-curve library and different distinct feature are extracted from the SFC images for malware detection. The classification is done using the convolutional neuronal network architecture. All the 24 families of malware are classified using the CNN model and a confusion matrix is made to differentiate and identify the families of malwares.

CHAPTER 3

LITERATURE SURVEY

In this section, we review related visualisation-based techniques applied to malware analysis. Our focus is on binary-based approaches where malware binary programs are converted into image format and from there, discriminant features are extracted for malware detection. Such approaches do not require any preliminary pre-processing of the malware binaries to extract characteristic features.

3.1 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 un-encoded 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 byteplots for malware classification.

3.2 Byte-level malware classification based on markov images and deep learning:

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 gray-scale 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.

3.3 A survey on machine learning-based malware detection in executable files:

Malware and benign samples are analyzed using static or dynamic analysis techniques. After analysis unique features are extracted to distinguish the malware and benign files. The efficiency of the malware detection system depends on how effectively discriminative malware features are extracted through the analysis techniques. There are various methods to set up the analysis environments using various static and dynamic tools. The second phase is to train the malware classifiers. Earlier traditional methods were used but nowadays machine learning algorithms are used for malware classification which can cope with complexity and pace of malware development. In this paper detailed study of malware detection techniques using machine learning algorithms are presented. In addition, this paper discusses various challenges for developing malware classifiers. At last future directive is discussed to develop an effective malware detection system by handling various issues in malware detection. This research study contributes to malware detection using runtime features. It shares the fraction in the body of knowledge for malicious software analysis and detection techniques using machine learning algorithms. Nowadays, it is well understood that malware are very complex and developing at so high rate. Today these are not only used to annoy the user, steal data, destroy user's data but also they are being used by organization or countries for fulfilling their objective. In the field of computer science, security is much-needed ingredients to secure the data and resources. It is so important because the computer is used in every place or embedded digital equipments to perform the task fast and more accurately without human attraction. Because of this large scope, it makes the computer system more vulnerable to get compromised by attackers. This research study presents the evolution, present state and detection technique for

malware. Malware are analyzed using static and dynamic techniques. Nowadays, malware are very specific and complex to harm the computer system by destroying computer resources or to steal the personal and confidential information of users. For developing a malware detection system, malware samples are analyzed to get the features which can represent the maliciousness. The principle of malware analysis is to extract useful information from the malware and use that information to detect and classify the malware. Two malware analysis techniques static and dynamic malware analysis techniques are used. Based on the analysis type, signature-based (antivirus software) and behaviour-based anti-malware systems are developed. Signature-based techniques suffer from two big problems. Firstly, new and unknown malware cannot be detected by signature-based techniques. Secondly, variants of malware can easily overrun the detection systems. While behaviour-based techniques can detect the new and variants of malware, additionally dynamic techniques are more robust to handle the malware obfuscation techniques than signature-based techniques. However, the practical implementation of the dynamic technique is very inflexible and time consuming whereas signature-based techniques are fast and proficient to detect the known malware than dynamic techniques.

3.4 Malware Detection Using Machine Learning:

In this paper, we present a framework for malware detection aiming to get as few false positives as possible, by using a simple and a simple multi-stage combination (cascade) of different versions of the perceptron algorithm. Other automate classification algorithms could also be used in this framework, but we do not explore here this alternative. The main steps performed through this framework are sketched as follows:

- A set of features is computed for every binary file in the training or test datasets, based on many possible ways of analyzing a malware.
- A machine learning system based firstly on one-sided perceptrons, and then on feature mapped one-sided perceptrons and a kernelized one-sided perceptrons, combined with feature selection based on the F1 and F2 scores, is trained on a medium-size dataset consisting of clean and malware

files. Cross-validation is then performed in order to choose the right values for parameters. Finally, tests are performed on another, non-related dataset. The obtained results were very encouraging.

- In the end we will analyse different aspects involved in the scale-up of our framework to identifying malware files on very large training datasets.

The main target was to come up with a machine learning framework that generically detects as much malware samples as it can, with the tough constraint of having a zero false positive rate. We were very close to our goal, although we still have a non-zero false positive rate. In order that this framework to become part of a highly competitive commercial product, a number of deterministic exception mechanisms have to be added. In our opinion, malware detection via machine learning will not replace the standard detection methods used by anti-virus vendors, but will come as an addition to them. Any commercial anti-virus product is subject to certain speed and memory limitations, therefore the most reliable algorithms among those presented here are the cascade one-sided perceptron (COS-P) and its explicitly mapped variant (COS-PMap). Since most AntiVirus products manage to have a detection rate of over 90%, it follows that an increase of the total detection rate of 3% – 4% as the one produced by our algorithms, is very significant. (Note that the training is performed on the malware samples that are not detected by standard detection methods.) As of this moment, our framework was proven to be a valuable research tool for the computer security experts at Bit Defender Anti Malware Research Labs. For the near future we plan to integrate more classification algorithms to it, for instance large margin perceptrons and Support Vector Machines.

3.5 Empirical Analysis of Learning-based Malware Detection Methods using Image Visualization:

In this research work, they implement the Image visualization-based malware detection method and conduct an empirical analysis of various learners for selecting a candidate learning classifier that can provide better prediction performance. Evaluate the framework using the following malware datasets, Search And Retrieval of Malware (SARVAM), Xue-dataset, and Canadian

Institutes for Cyber Security (CIC) datasets. The experiments include the following learning algorithms, Linear Regression, Random Forest, K-Nearest Neighbour (KNN), Classification and Decision Tree (CART), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and deep learning-based Convolutional Neural Network (CNN). This image-visualization-based method proves to be effective in terms of prediction accuracy. The CNN model achieved a high performance of F1-score and accuracy in the binary classification task reaching 95.70% and 99.50%, consecutively. The model in the multi-classification task achieved of 95.96% and 99.30% (F1-score and accuracy) for detecting malware types.

In this work, they implemented the computer visualization based 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.

3.6 Hybrid image transformation for malware classification:

In this article, they propose a novel approach to encoding and arranging bytes from binary files into images. These developed images contain

statistical (eg, entropy) and syntactic artifacts (eg, strings), and their pixels are filled up using space-filling curves. Thanks to these features, our encoding method surpasses existing methods demonstrated by extensive experiments. In particular, the proposed method achieved 93.01% accuracy using the combination of the entropy encoding and character class scheme on the Hilbert curve.

3.7 Identification of malicious code variants based on image visualization:

In the paper , they proposed the use of collective local and global malicious patterns for efficient malware variant identification. To reduce the computational time, the total number of dimensions of both types of patterns is reduced using selection methods. In addition, they prepared a baseline to compare the classification performance of the proposed model with previous state-of-the-art malware detection techniques. The experimental results indicate that the response time and classification performance of the model are better than those of previous models.

3.8 Image-based malware classification hybrid framework based on space-filling curves:

This paper presents a hybrid framework for malware classification designed to overcome the challenges incurred by current approaches. The framework incorporates novel static and dynamic malware analysis methods, where static malware executables and dynamic process memory dumps are converted to images mapped through space-filling curves, from which visual features are extracted for classification. The framework is 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. On a dataset of 13,599 obfuscated and non-obfuscated malware samples from 23 families, the framework outperformed both static and dynamic standalone methods with precision, recall and accuracy scores of 97.6%, 97.6% and 97.6% respectively.

3.9 Malware analysis using visualized images and entropy graphs:

This paper proposes a new malware family classification method by converting binary files into images and entropy graphs. The experimental results show that the proposed method can effectively distinguish malware families.

3.10 Study of a Hybrid Approach Towards Malware Detection in Executable Files:

This paper proposes a CNN-based model that extracts visual features from malware images, which outperforms existing approaches on a benchmark dataset like Malimg. We study the effect of using a hybrid feature set containing these visual features integrated with statically obtained opcode frequencies for the detection of malware. Our experiments on standard datasets demonstrate that there is no significant performance improvement using this hybrid approach. Classical signature-matching mechanisms used by most AVs tend to become unreliable when they find new malware. Towards this endeavor, we utilized a hybrid feature set obtained from the discriminatory feature of PE files, opcode frequency, and high-level image descriptors from a trained CNN model for the detection of malicious code in the Windows executable files. However, we did not observe any significant improvement over the opcode frequency feature set, probably due to an increase in dimensionality, but we observed a drastic change in accuracy when the images were used for the detection of malware. In future, we will study the effect of integrating other features such as opcode sequence calls information in more effective manner. We also extend this approach to classify malware into their corresponding families once detected, and further processes for better accuracy. Our study can guide future researchers who plan to use different data sources for malware analysis.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS:

Hard Disk : 120GB

RAM : 8 GB

Devices : Monitor , Keyboard , Mouse.

4.2 SOFTWARE REQUIREMENTS:

Operating System : Windows 10

Software : VMWare WorkStation

Coding language: Python

CHAPTER 5

DESCRIPTIONS OF PROPOSED SYSTEMS

5.1 EXISTING SYSTEM:

Various researchers have proposed ways to build advanced malware detectors that can mitigate threat actors and enable effective cybersecurity decisions in the past. Recent research implements malware detectors based on visualized images of malware executable files. In this framework, a malware binary is converted into an image, and by extracting image features and applying machine learning methods, the malware is identified based on image similarity. In this research work, we implement the Image visualization-based malware detection method and conduct an empirical analysis of various learners for selecting a candidate learning classifier that can provide better prediction performance. We evaluate our framework using the following malware datasets, Search And Retrieval of Malware (SARVAM), Xue-dataset, and Canadian Institutes for Cyber Security (CIC) datasets. Our experiments include the following learning algorithms, Linear Regression, Random Forest, K-Nearest Neighbour (KNN), Classification and Decision Tree (CART), Support Vector Machine (SVM), Multi Layer Perceptron (MLP), and deep learning-based Convolutional Neural Network (CNN). This image-visualization-based method proves to be effective in terms of prediction accuracy. Some conclusions emerge from our initial study and find that a Convolutional Neural Network (CNN) algorithm provides relatively better performance when used against SARVAM and various malware datasets. The CNN model achieved a high performance of F1-score and accuracy in the binary classification task reaching 95.70% and 99.50%, consecutively. The model in the multi classification task achieved of 95.96% and 99.30% (F1-score and accuracy) for detecting malware types. We find that the KNN model outperforms other traditional classifiers.

5.2 PROPOSED SYSTEM:

The framework incorporates novel static and dynamic malware analysis methods. Static malware executables and dynamic process memory dumps are converted to images mapped through space-filling curves, from which the visual features are extracted for malware detection. Image-based approaches to malware classification involve the conversion of malware binaries to 2-dimensional images, such that characteristic properties or features of the original file are retained. Computer vision techniques are then used to extract discriminant features from the images for malware detection.

The framework combines static and dynamic features gathered from malware samples converted to 2-dimensional images, mapped through space-filling curves (SFC) traversals. Only the 2-dimensional is considered since the outputs are 2-dimensional images. The 2-dimensional unit square represents an image of $n \times n$ pixels. The SFCs pass through every pixel point of a regular spatial region. Spatial locality of the data is preserved i.e. closely located points in the binary file space will also be closely located when mapped to an SFC image.

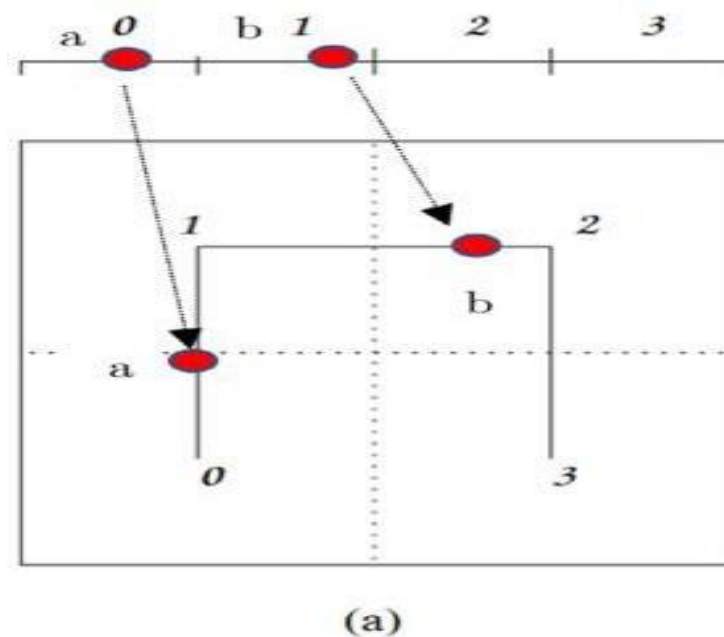
The data structure within the original malware binaries can be retained in the resulting SFC images. This method processes non-obfuscated samples statically and obfuscated samples dynamically. Non-obfuscated samples do not require any behavioural dynamic analysis because sufficient features can be captured from static binary. Dynamic analysis captures the behavioural data of the dynamic data. The features which are extracted are used for malware detection process.

5.2.1 *Space-filling curves:*

The locality preservation properties of SFC's mean they can be utilised in many areas of computing, such as partitioning or reordering data and computations (Moon et al., 2001). Practical applications of SFC implementations can be found in technologies such as database indexing (Huang, 2013), parallel computing Nivarti et al. and image retrieval (Nguyen et al., 2012). Google's S2 Geometry library uses the Hilbert curve for Geo-spatial indexing in the Google Maps application (S2-Geometry, 2017). In the context of this research, the aim is to capture all relevant characteristic features of malware binaries in an image format

so that discriminant features may be extracted for classification purposes. At the core of these techniques is a layout function that inputs a set of values and outputs a partition of space on the resulting mapped image, such that regions represent disjoint data that do not intersect. Previously, tree-maps, have been presented as a solution to the graphical display of data (Johnson and Shneiderman, 1991). However, for high dimensional, dense layout tree maps, aspect (width-to-height) ratios tend to be extremely high and thus elements in the mapped image are difficult to correlate and interpret. Wattenberg (2005) defined the notion of a “perfect” layout function, i.e., a layout method for a space-filling visualization that satisfies the following four desirable criteria:

- **Stability:** a small change in inputs should produce a subsequent small change in output.
- **Split neutrality:** any structural changes do not affect the structure of other regions on the image map.
- **Order adjacency:** each similar item should be located adjacent to each other;
- **Locality:** layout regions should be relatively compact. If layout regions are represented loosely, e.g. long and narrow, then the visualisation of regions will not be as clear.



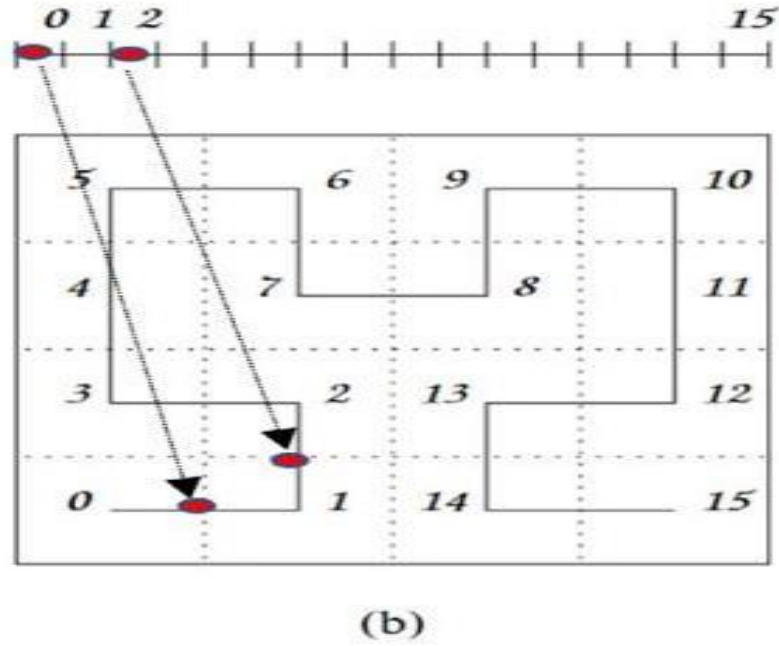


Fig.5.1: Mapping data through SFC curve. Points a and b represent data points mapped to the resulting 2-dimensional SFC image.

A proof that any perfect layout function is actually an instance of a jigsaw map, so-called due to its resemblance of a jigsaw puzzle, thus providing a complete characterization of perfect layouts in terms of screen-filling curves. In particular, SFCs possess these four essential properties, therefore are a practical solution to representing data in a 2-dimensional format.

The aim of this evaluation was to investigate the influence of varying sizes of SFC images on classification performance and time complexity. The objective was to choose an image size that was feasible in terms of its conversion and processing times, while returning optimum classification performance. The dimensions of an image can influence the number of features produced, which can in turn have an effect on classification performance and processing speed. Our aim was to build a robust framework that was also scalable, so time complexity was an additional consideration. The evaluation was conducted on images ranging from 32×32 pixels to 512×512 pixels. Processing times for image sizes above 512 became infeasible for both conversion and classification and so are not considered. The influence of the varying image dimensions on classification performance. From the results, it is evident that as image dimension size increases, performance improves, but at a cost of slower processing times. For the purposes of our further analysis,

we chose the 256×256 dataset, since it yielded a high classification performance, 97.7% across all three metrics, within a reasonably fast processing time, at 8ms per sample. The 512×512 dataset yielded marginally better classification results, but processing time was 3.5 times slower. Furthermore, the conversion process from binary to 512×512 image, took approximately 5 seconds per sample, which would not be scalable to large datasets.

5.3 ARCHITECTURE:

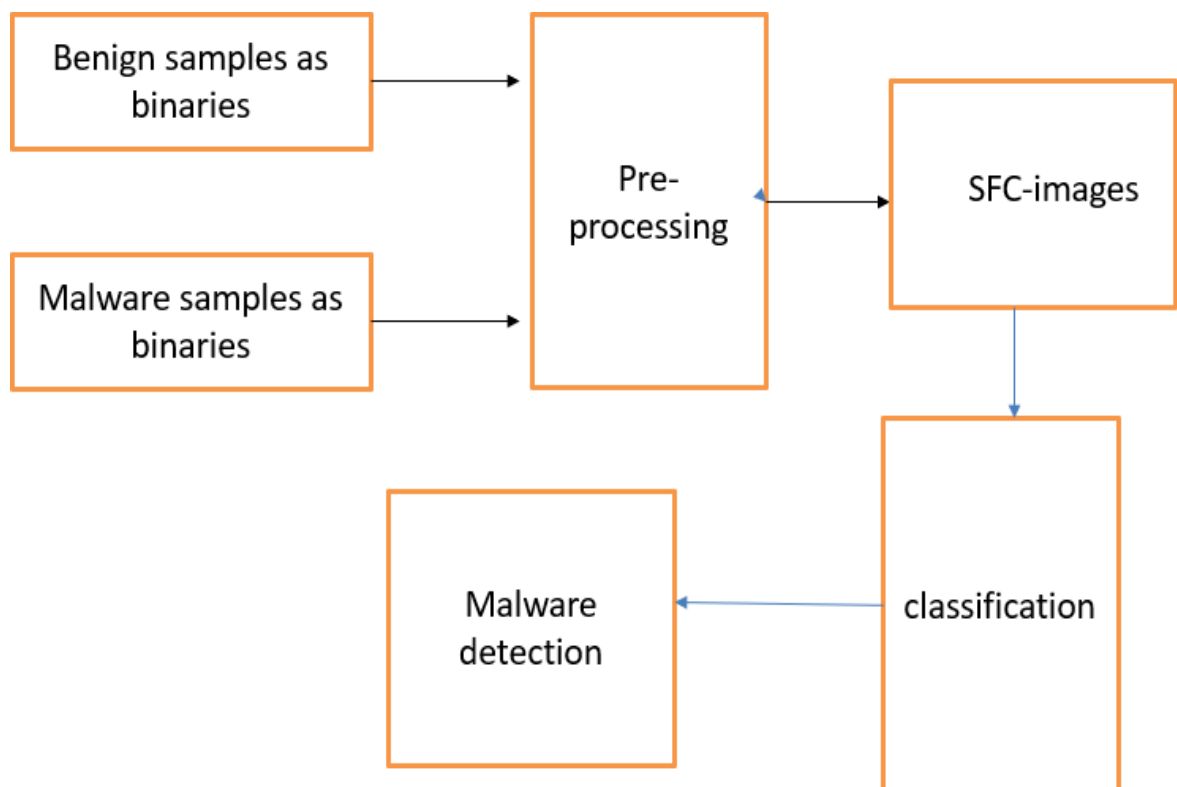


Fig 5.2: Architecture of malware detection and classification.

5.4 MODULES

- Data gathering and pre-processing
- Mapping binary files
- Classification
- Malware detection

5.4.1 DATA GATHERING AND PREPROCESSING:

A dataset comprising 32-bit executable portable executable (PE) samples, from 23 distinct malware families was compiled from the Virus Total Academic share repository. Three image datasets were derived from the samples through the Hilbert, Graycode and Z-order curve traversals using the scurve library. Out of the 23 family classes, 5 were considered obfuscated. The dataset contains a more up-to-date set of samples than the Malimg and Kaggle datasets, with some of the most prevalent and current malware species in circulation today. To test the predictive capabilities of the classification models on unseen malware, an evaluation set of samples or approximately 10% of the data was held out from the original dataset. Samples were chosen using stratified random sampling, which gives a subset structure that is representative of the entire dataset. Prior to conversion of the malware binaries to SFC image format, the Shannon entropy for each sample was first calculated to determine the level of obfuscation, if any. Shannon entropy measures the degree of randomness for a given set of data, ranging from 0 for orderly or non-random data, to 8 for very random data. For binary files, Shannon entropy can be used as an indicator for malware obfuscation since techniques such as encryption or packing distort binary data and therefore produce a high entropy score. The average entropy for packed malware binaries was found to be 6.8, whereas the average entropy for encrypted binaries was 7.1. The average entropy for a native executable file was found to be 5.1. For the purposes of this research, we considered the value μ as the Shannon entropy threshold of 6.8, to determine if samples were obfuscated.

Prior to conversion of the malware binaries to SFC image format, the Shannon entropy for each sample was first calculated to determine the

level of obfuscation, if any. Shannon entropy measures the degree of randomness for a given set of data, ranging from 0 for orderly or non-random data, to 8 for very random data. For binary files, Shannon entropy can be used as an indicator for malware obfuscation since techniques such as encryption or packing distort binary data and therefore produce a high entropy score. Previous work by Lyda and Hamrock (2007) demonstrated that obfuscation techniques such as encryption and packing increase the entropy value of the original binary file. Using a dataset of portable executable malware files, the authors produced a table of entropy ranges to identify the obfuscation type. The average entropy for packed malware binaries was found to be 6.8, whereas the average entropy for encrypted binaries was 7.1. The average entropy for a native executable file was found to be 5.1. For the purposes of this research, we considered the value μ as the Shannon entropy threshold of 6.8, to determine if samples were obfuscated.

As stated previously, we optimised the framework by only extracting dynamic or behavioural data from the samples considered to be obfuscated, above the threshold μ . Dynamic analysis can be considered more invasive than static methods since the samples must be executed in a controlled environment to capture their behavioural data, which is then written to the physical host. Consequently, the time complexity to execute every sample in this manner would make it infeasible for large datasets. Furthermore, to implement this method on non-obfuscated malware samples would not yield any greater results, as demonstrated by O'Shaughnessy (2019), where classification performance for low entropy malware executables in SFC format was found to be high. For dynamic analysis, we implemented Virtual Machine Introspection (VMI), a process introduced by Garfinkel and Rosenblum (2003) that refers to the inspection or monitoring of operating systems residing inside a virtual machine from the hypervisor level. In this case, malware is passed to a virtual machine, is executed and a process memory dump is extracted using the Microsoft utility Process dump (Russovich and Richards, 2020). The dump file is then written back to disk on the host machine. To optimise the analysis process, mini process dumps containing process thread and handle information, were extracted from each malware in execution. While full process dumps contain the full malware process data, it was found they ranged on

average from 50 to 70 MB for each dump, which would not scale well for large datasets and so were not considered for the purposes of this research.

In this section we analyse the contents of sample process dump files in order to demonstrate the advantage of using dynamic data over static analysis for obfuscated malware. Static methods are limited to the analysis of structure and code of the malware binary. However, obfuscation methods such as packing, encryption and encoding alter the binary structure of the code to the extent that common salient features cannot be obtained for the purposes of classification. By executing malware in a controlled virtual environment, the binary must be de-obfuscated and written to memory. Thus, we can extract behavioural features characteristic of same family classes. As stated previously, we use the small or mini dump format for processing efficiency. Mini dumps contain the following data:

- The Stop message and its parameters and other data
- A list of loaded drivers
- The processor context (PRCB) for the processor that stopped
- The process information and kernel context (EPROCESS) for the process that stopped
- The process information and kernel context (ETHREAD) for the thread that stopped
- The Kernel-mode call stack for the thread that stopped

To illustrate the suitability of process dumps for representing malware behavioural features, we focus here on examining the call stack traces taken from two variants of the Chir family. The call stack is made up of the chain of function calls that have led to the current location of the program counter, where the process has been terminated by the Process dump tool. Intuitively, malware binaries that share similar source code will behave the same when executed on the target machine, which results in similar execution traces in the same sequence. These similarities can be used as signatures to group malware variants into their respective family classes. It is infeasible to show a complete stack trace of each program, so we confine our comparison to the process injection routine of the Chir samples, shown in Fig. 5. Chir malware implements a process injection routine to obfuscate its presence on the operating system. During this routine, the malware injects a copy

of itself, via svchost.exe, into WerFault.exe, which is a legitimate Windows program for error fault reporting. This specific routine is characteristic of the Chir variant, so could be considered as part of the signature for identification. The function calls and call sequences are identical, which indicates the two samples share the same behavioural traits for this code injection routine. The behavioural data captured from the running process dumps allowed us to overcome the obfuscation limitations and improve classification performance over the static standalone method.

A Malware can belong to one of the following class :

- Adialer.C
- Agent.FYI
- Allapple.A
- Allapple.L
- Alueron.gen!J
- Autorun.K
- C2LOP.P
- C2LOP.gen!g
- Dialplatform.B
- Dontovo.A
- Fakerean
- Instantaccess
- Lolyda.AA1
- Lolyda.AA2
- Lolyda.AA3
- Lolyda.AT
- Malex.gen!J
- Obfuscator.AD
- Rbot!gen
- Skintrim.N
- Swizzor.gen!E
- Swizzor.gen!I
- VB.AT
- Wintrim.BX
- Yunex.A

5.4.2 Mapping binary files:

A SFC traces a continuous curve through every unit square, i.e. pixel, in the image. To envision this for the purpose of file mapping, the binary code of a malware file can be thought of as a flattened out 1-dimensional line because the code is parsed sequentially, byte-by-byte. The SFC maps each point (bytes) from the 1-dimensional space to the 2-dimensional space (SFC image map) such

that closely located points in the binary file space will tend to also be closely located when mapped to the SFC. depicts the mapping of a 1-dimensional set of points to a 2-dimensional curve representation, in this case the Hilbert curve, in first and second order iterations. It shows that closely located data points (a and b in the example) map to similar positions on the SFC map. For the purposes of this research, we used s-curve, a library written in Python, to convert the malware binaries into SFC image format (Cortesi, 2012). The scurve library uses a colour-coding scheme, based on data type, that maps the malware binary data to the resulting SFC images. The colour scheme classifies bytes (0–255 range) into the following categories: black for 0×00 (0), blue for ASCII text (1–126), red for high-value bytes (127–254), and white for $0 \times ff$ (255). Fig. 3 shows an example of an SFC image in Z-order generated from the Chir family, which is a type of email worm file infector. The image was generated using the static PE executable file and processed with the scurve library. If the data type from the malware binary is distinct, it is displayed as non-overlapping sections of a single colour. displays a large region of black indicating large sections of the file contain zeros or null padding, which is sometimes used as a simple obfuscation method by malware developers. There is also a large region of printable characters, shown in blue with some small sections of high value bytes, denoted by the red regions. The rest of the file comprises a mixture of multiple data types, which give the resulting image distinct texture regions. Distinct features that are common to related species of malware can be extracted from the textures produced in the SFC images for classification

Once the malware samples have been processed, the next stage in the framework is to convert the non-obfuscated malware executable samples and the obfuscated sample process dumps to SFC format through the scurve library. This section discusses the implementation of the mapping procedures using the scurve library outlined. The Hilbert implementation differs from the Z-order and Gray-code methods, which are inherently similar. The Hilbert implementation first determines the dimensions of input binary. It then iterates through each coordinate point, which is mapped to its data colour class. The implementation of the Z-order and Gray code curves are similar, in that the scurve library considers the bit range for each distinct chunk of binary data as coordinates. In other words, the start offset is the first coordinate and the end offset is the second coordinate. In the case of the Z-order

code, the bit interleaving process then takes place. For the Gray code, the coordinates are XORed from left to right. The resulting codes in each case, represent the cell that the binary chunk will occupy, which is mapped to a distinct colour, depending on the data colour class. An individual sample taken from the Allapple malware family, transformed into SFC images through the Hilbert, Gray-code and Z-order traversals. The areas of blue (ASCII printable characters) and the smaller areas of red (high value non-printable bytes) are evidently similar across the three SFC images. However, the positions of the texture patterns differ, due to the different traversal patterns of the space-filling curves. Examples of SFC images converted from the process dump files. Each image represents a variant of the Cryptowall ransomware family. The Cryptowall variant uses encrypted sections as a means of obfuscation and so static analysis methods produce poor results. This necessitated their execution through VMI to gather behavioural data for classification. It can be seen from the images that the data texture regions are sparse, with large regions of black, which equate to binary zeros. An examination of the memory dumps confirmed that the majority of the files comprised of binary zeros.



Fig no:5.3 The converted image from the binary data of the PE files.

5.4.3 Malware classification:

Three feature descriptor algorithms were evaluated, namely Local Binary Patterns (LBP) (Ojala et al., 1996), Gabor filters (Gabor, 1946) and Histogram of Oriented Gradients (HOG) (McConnell, 1986). While the HOG descriptor is primarily used for object detection in images, it has produced favourable results in texture analysis (Demir, 2018; Harb et al., 2017). 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 3×3 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, proposed by Ojala et al. (2000). 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 neighborhood 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. a schematic of the LBP feature extraction process used in this research.

The SFC image is first converted to grayscale. Next, the LBP operator is applied which returns a histogram of features that is constructed from the LBP number values. Next, each histogram is added to an array. As each histogram is calculated, it is concatenated to the array. The result is an array of feature histograms, which represent the feature vector for the image. Gabor filters The Gabor filter is a sinusoidal signal of a given frequency and orientation. Since a single filter is spatially localised, banks of Gabor filters with varying orientations and frequencies are applied to the image as the filter must pass in the same direction as the target texture to return accurate features. In our descriptor we used a Gabor bank of 0–

180 at intervals of 45 to ensure we captured sufficient texture features from varying directions.

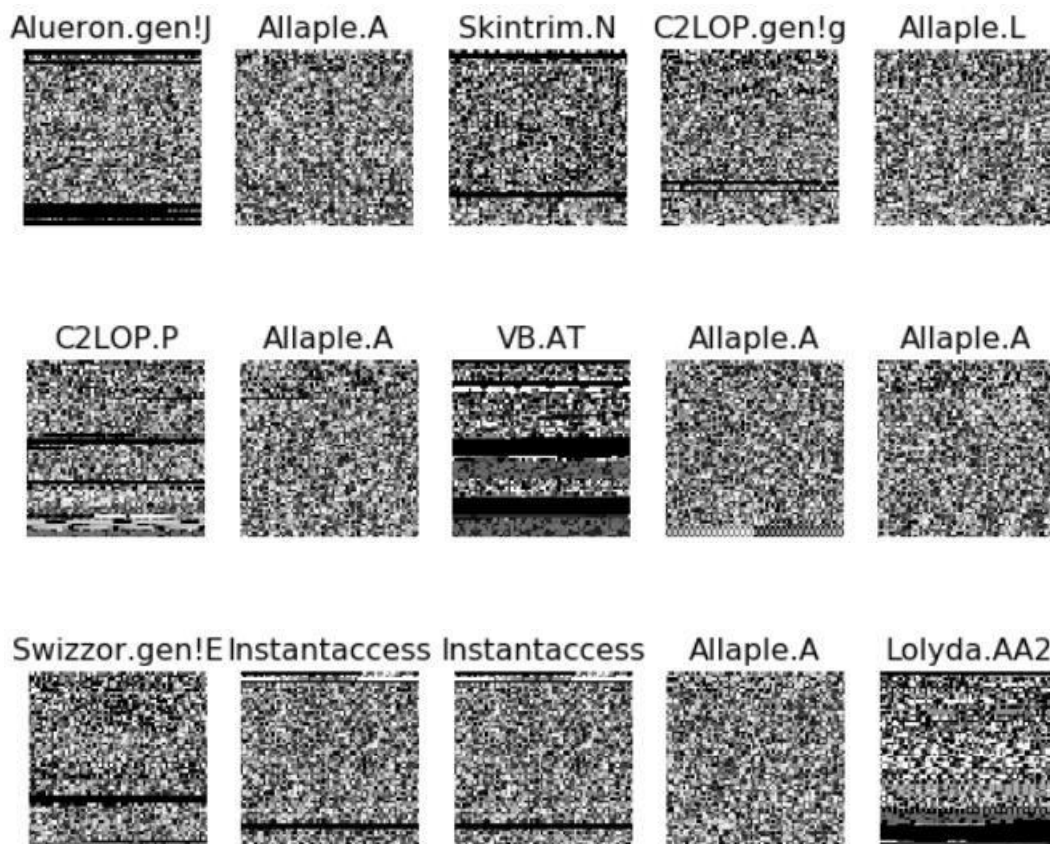
The main Gabor filter parameters considered in this work were:

- k : is the size in pixels of the Gabor kernel. The value of k is preferably odd and the kernel is a square for the sake of uniformity.
- σ : is the standard deviation of the Gaussian function used in the Gabor filter.
- θ : is the orientation of the normal to the parallel stripes of the Gabor function.
- γ : represents the spatial aspect ratio depicts the Gabor filter process used in this research.

The example image is taken from the Emotet banker Trojan family in Hilbert format. The SFC image is input and a series of filters are applied in varying orientations and frequencies. The resulting filters are then convoluted to produce the final feature vector for the image, which is appended to the feature array. Histogram of oriented gradients The Histogram of Oriented Gradients (HOG) feature descriptor counts occurrences of gradient orientation in localized portions of an image. Local object appearance and shape within an image can be characterized by the distribution of intensity gradients or edge directions. Fig. 10 shows the HOG feature extraction process used in our framework. The source SFC image is divided into blocks and each block is sub-divided into cells. The vertical and horizontal gradients are computed for each pixel in a cell. The orientation histogram is then created for each cell. Bins represent the different angles of orientation. The final feature descriptor consists of a concatenation of the cell histograms for each block, which is then flattened into a feature vector. The parameters considered when tuning the HOG descriptor were orientation, pixels-per-cell and cells-per-block. The orientation parameter represents the number of orientation bins in the resulting histograms. Pixels-per-cell determine the size, in pixels, of each cell. Similarly, cells-per-block determine the size of each block. It should be noted here that the optimum parameters for each descriptor were determined through an extensive tuning process.

The next stage in the framework is the classification stage. The aim of classification in this context is to order the malware samples into their taxonomic groups or family classes, based on the textural similarities extracted from the SFC images by the

LBP, Gabor and HOG descriptors. Supervised learning algorithms were used exclusively in this research since the class labels, i.e., the malware family names, were already known. The trained classification models were then tested on the 10% holdout validation set, described , to gauge how well they generalised to previously unseen data. The following classification algorithms were evaluated: Random Forest (RF), Support Vector Machine (SVM) and K-nearest Neighbours (KNN). These classifiers were chosen as previous research by O'Shaughnessy (2019) demonstrated favourable performance results when applied to malware SFC feature datasets. Efforts were made during the classification training phases to reduce the possibility of underfitting and overfitting in the resulting classification models. Overfitting occurs when the model fits the data too well, mainly due to it capturing noise along with the underlying signatures or patterns in the data. A model that is trained to fit slightly inaccurate data can infect it with substantial errors and reduce its predictive power. To reduce the risk of underfitting, a suitably large dataset was compiled. Stratified k-fold cross-validation was used to minimize overfitting.



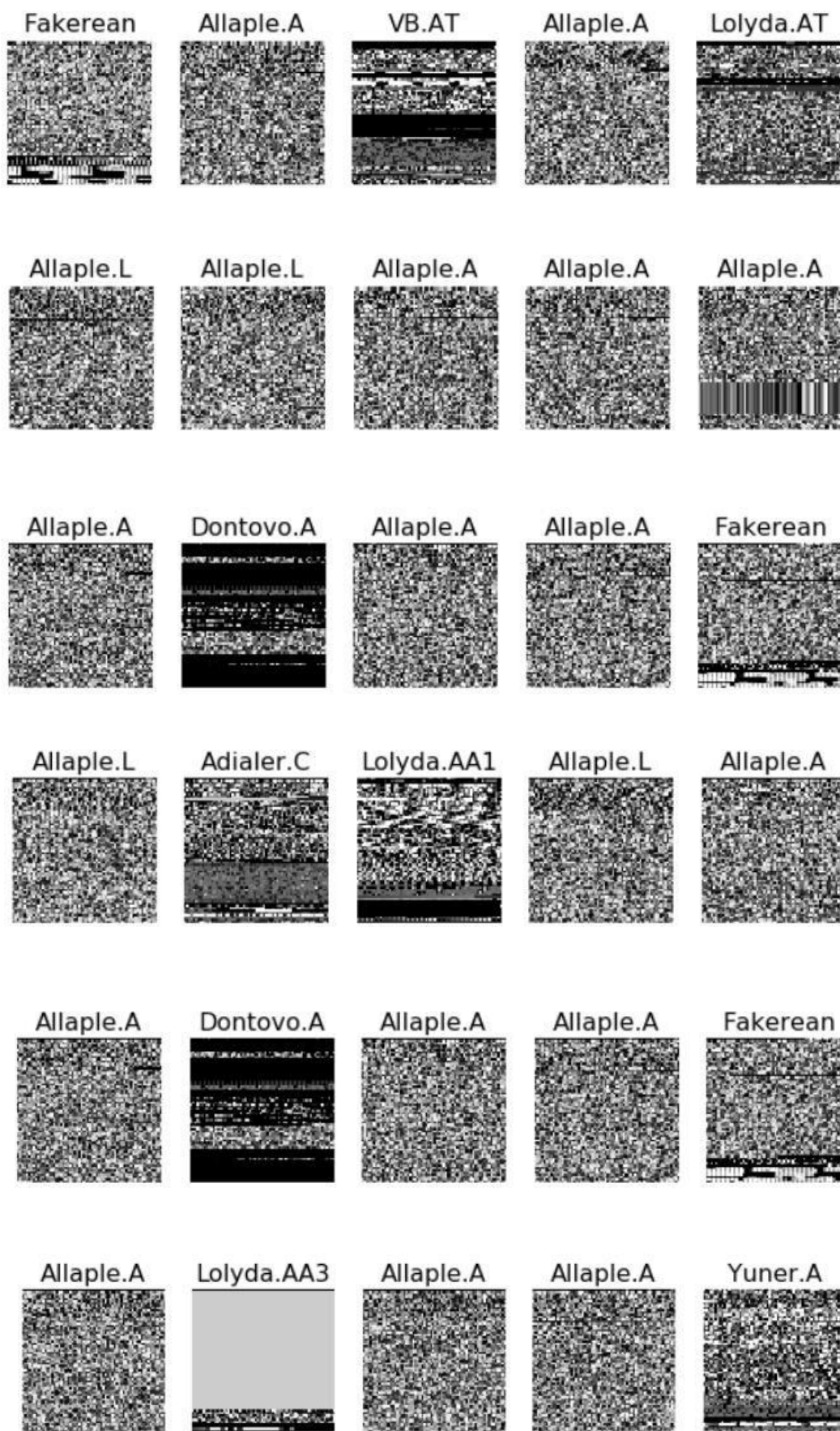




Fig no:5.4 The different families of malwares.

CNN ALGORITHM

Convolutional Neural Network is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category 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, 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 Soft-max function to classify an object with probabilistic values 0 and 1.

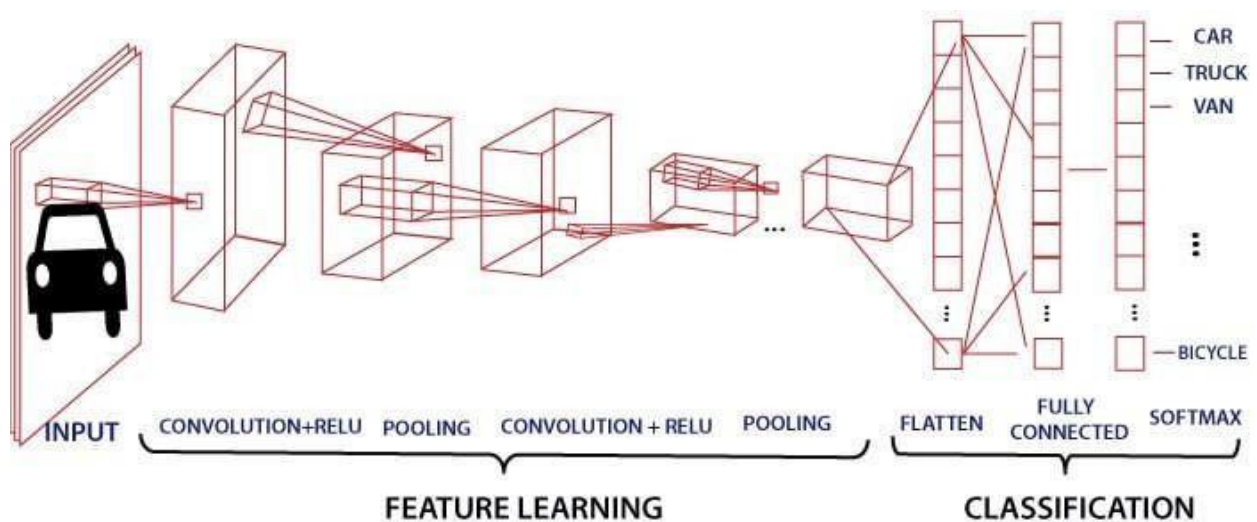


Fig No. 5.5 CNN based Soft-max Function

We will now build our **CNN** model using Keras.

This model will have the following layers :

- **Convolutional Layer** : 30 filters, (3 * 3) kernel size
- **Max Pooling Layer** : (2 * 2) pool size
- **Convolutional Layer** : 15 filters, (3 * 3) kernel size
- **Max Pooling Layer** : (2 * 2) pool size
- **DropOut Layer** : Dropping 25% of neurons.
- **Flatten Layer**
- **Dense/Fully Connected Layer** : 128 Neurons, Relu activation function
- **DropOut Layer** : Dropping 50% of neurons.
- **Dense/Fully Connected Layer** : 50 Neurons, Softmax activation function
- **Dense/Fully Connected Layer** : num_class Neurons, Softmax activation function

Input shape : 64 * 64 * 3

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 30)	840
max_pooling2d_1 (MaxPooling2)	(None, 31, 31, 30)	0
conv2d_2 (Conv2D)	(None, 29, 29, 15)	4065
max_pooling2d_2 (MaxPooling2)	(None, 14, 14, 15)	0
dropout_1 (Dropout)	(None, 14, 14, 15)	0
flatten_1 (Flatten)	(None, 2940)	0
dense_1 (Dense)	(None, 128)	376448
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 50)	6450
dense_3 (Dense)	(None, 25)	1275
=====		
Total params: 389,078		
Trainable params: 389,078		
Non-trainable params: 0		

Fig no : 5.6 Classified malware families

5.4.4 Malware detection:

By using the features extracted 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 identified to which family it belonged to using the convolutional neural networks.

CHAPTER 6

RESULTS AND CONCLUSIONS

We 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 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, we intend 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.

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. The malware families are identified using CNN algorithm.

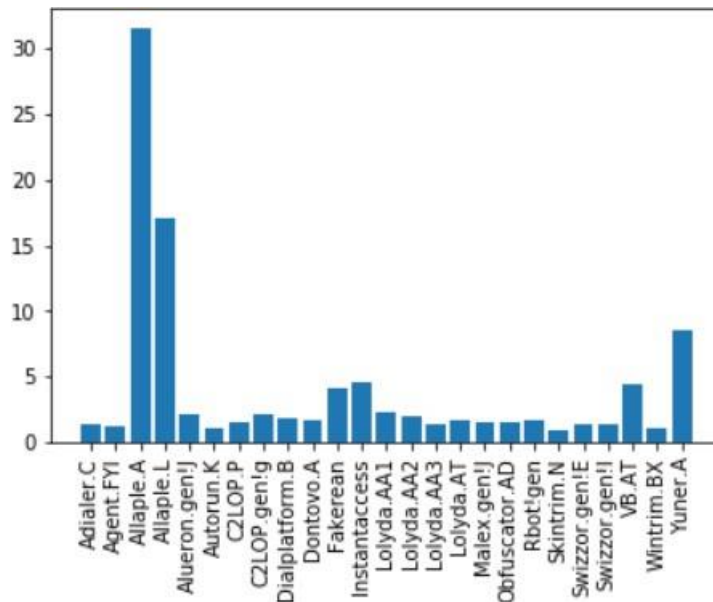


Fig no:6.1 Bar graph of malware families.

Train on 6537 samples, validate on 2802 samples

```
Epoch 1/10
6537/6537 [=====] - 97s 15ms/step - loss: 1.6495 - acc: 0.4999 - val_loss: 0.7917 - val_acc: 0.7409
Epoch 2/10
6537/6537 [=====] - 95s 15ms/step - loss: 0.6894 - acc: 0.7846 - val_loss: 0.3123 - val_acc: 0.9115
Epoch 3/10
6537/6537 [=====] - 96s 15ms/step - loss: 0.3987 - acc: 0.8704 - val_loss: 0.2241 - val_acc: 0.9261
Epoch 4/10
6537/6537 [=====] - 93s 14ms/step - loss: 0.2838 - acc: 0.9117 - val_loss: 0.1876 - val_acc: 0.9450
Epoch 5/10
6537/6537 [=====] - 96s 15ms/step - loss: 0.2290 - acc: 0.9257 - val_loss: 0.1748 - val_acc: 0.9479
Epoch 6/10
6537/6537 [=====] - 93s 14ms/step - loss: 0.2028 - acc: 0.9338 - val_loss: 0.1494 - val_acc: 0.9582
Epoch 7/10
6537/6537 [=====] - 74s 11ms/step - loss: 0.1914 - acc: 0.9397 - val_loss: 0.1363 - val_acc: 0.9561
Epoch 8/10
6537/6537 [=====] - 70s 11ms/step - loss: 0.1689 - acc: 0.9436 - val_loss: 0.1503 - val_acc: 0.9554
Epoch 9/10
6537/6537 [=====] - 69s 11ms/step - loss: 0.1622 - acc: 0.9449 - val_loss: 0.1336 - val_acc: 0.9579
Epoch 10/10
6537/6537 [=====] - 71s 11ms/step - loss: 0.1431 - acc: 0.9527 - val_loss: 0.1304 - val_acc: 0.9579
: <keras.callbacks.History at 0x173b8908630>
```

Fig no :6.2 Test and train of malware.

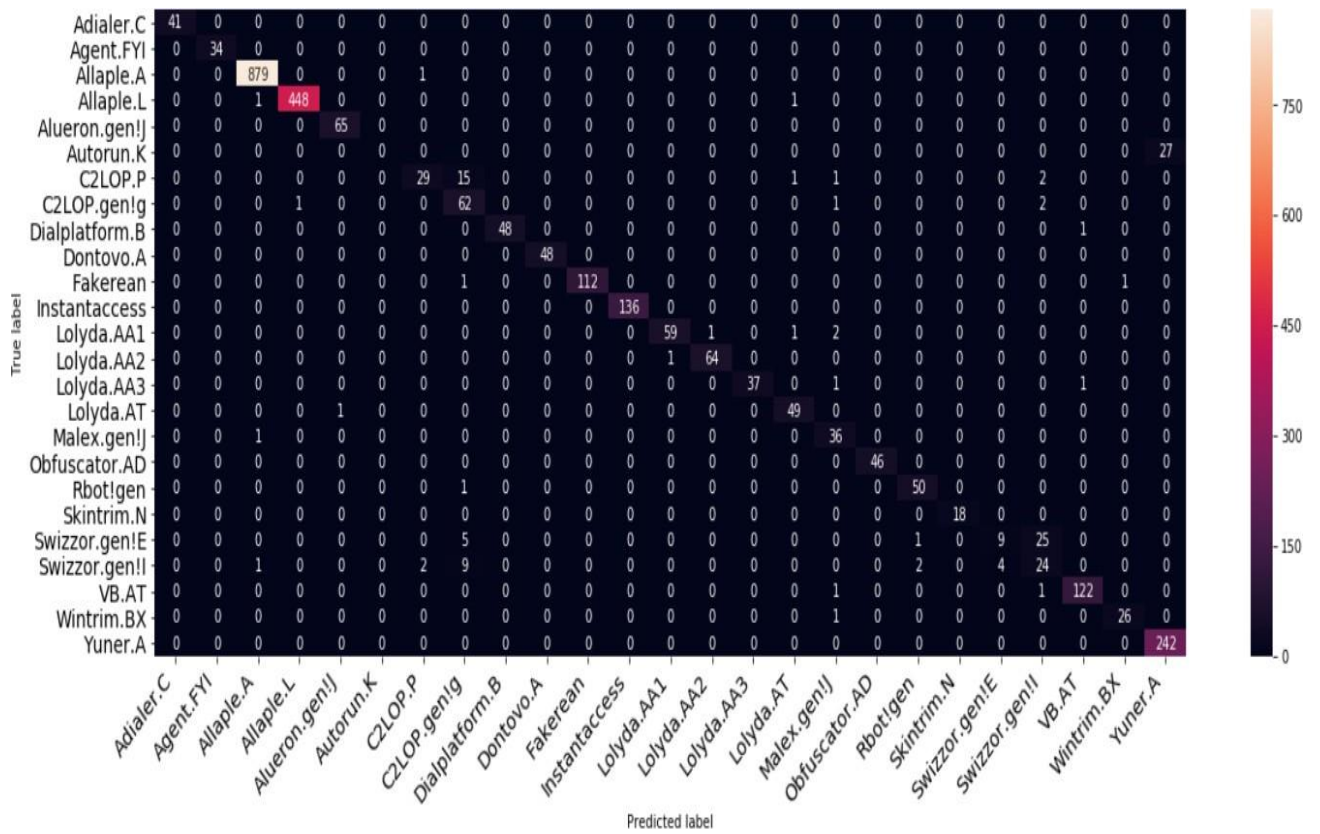


Fig no:6.2 confusion matrix

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.-L. (Eds.), *Visualization for Computer Security*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–17.
- [5] 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.
- [6] Ojala, T., Pietikäinen, M., Harwood, D., 1996. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognit* 29 (1), 51–59. doi:10.1016/0031-3203(95)00067-4.
- [7] Oliva, A., Torralba, A., 2001. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int J Comput Vis* 42 (3). doi:10.1023/A:1011139631724.
- [8] O'Shaughnessy, S., 2019. Image-based malware classification: a space filling curve approach. 2019 IEEE Symposium on Visualization for Cyber Security (VizSec) doi:10.1109/vizsec48167.2019.9161583.
- [9] ÓShaughnessy, S., 2019. Sfc datasets. <https://github.com/stephen-oshaughnessy/SFC-datasets>. Peano, G., 1890. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen* 36 (1), 157–160. doi:10.1007/bf01199438.

- [10] Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M., 2018. Microsoft malware classification challenge. arXiv:1802.10135.
- [11] Russinovich, M., Richards, A., 2020. Procdump - windows sysinternals. <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>. S2-Geometry, G., 2017. S2 cells. https://s2geometry.io/devguide/s2cell_hierarchy.html.
- [12] Shannon, C.E., 1948. A mathematical theory of communication. Bell System Technical Journal 27 (3), 379–423. doi:10.1145/584091.584093.
- [13] Sklearn, 2020. Sklearn gridsearchcv. https://scikit-learn.org/0.17/modules/generated/sklearn.grid_search.GridSearchCV.html. Vasan, D., Alazab, M., Wassen, S., Safaei, B., Zheng, Q., 2020.
- [14] Image-based malware classification using ensemble of cnn architectures (imcec). Computers & Security 92, 101748. doi:10.1016/j.cose.2020.101748.
- [15] Vu, D., Nguyen, T., Nguyen, T.V., Nguyen, T.N., Massacci, F., Phung, P.H., 2019. Hit4mal: hybrid image transformation for malware classification. Transactions on Emerging Telecommunications Technologies 31 (11). doi:10.1002/ett.3789.
- [16] Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D.A., Aigner, W., 2015. A survey of visualization systems for malware analysis.
- [17] In: Borgo, R. (Ed.), Eurographics Conference on Visualization (EuroVis) ; STARs - State of The Art Reports. The Eurographics Association, pp. 105–125. doi:10.2312/eurovisstar.20151114.
- [18] Wattenberg, M., 2005. A note on space-filling visualizations and space-filling curves. In: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005, pp. 181–186. doi:10.1109/INFVIS.2005.1532145.
- [19] Fukushima, Y., Sakai, A., Hori, Y., Sakurai, K., 2010. A behavior based malware detection scheme for avoiding false positive. In: 2010 6th IEEE Workshop on Secure Network Protocols, pp. 79–84. doi:10.1109/NPSEC.2010.5634444.
- [20] Gabor, D., 1946. Theory of communication. Journal of Institution of Electrical Engineers 93 (3), 429–457. doi:10.1049/ji-3-2.1946.0074.

- [21] Garfinkel, T., Rosenblum, M., 2003. A virtual machine introspection based architecture for intrusion detection. In: In Proc. Network and Distributed Systems Security Symposium, pp. 191–206. Gray, F., 1953. Pulse code communication.
- [22] Harb, H.M., Desuky, A.S., Mohammed, A., Jennane, R., 2017. Histogram of oriented gradients and texture features for bone texture characterization. *Int J Comput Appl* 165 (3), 23–28. doi:10.5120/ijca2017913820.
- [23] Hilbert, D., 1891. Über die stetige abbildung einer linie auf ein flächenstück. *Mathe matische Annalen Zeitschriftenband* (1891) 459—460. Huang, Y.-K., 2013. Indexing and querying moving objects with uncertain speed and direction in spatiotemporal databases. doi:10.1007/s10109-013-0191-6.
- [24] Johnson, B., Shneiderman, B., 1991. Tree-maps: a space-filling approach to the visu alization of hierarchical information structures. In: *Proceeding Visualization '91*, pp. 284–291. doi:10.1109/VISUAL.1991.175815.
- [25] Kaspersky, 2020. The number of new malicious files detected every day increases by 5.2% to 360,000 in 2020. Kaspersky.com. https://www.kaspersky.com/about/press-releases/2020_the-number-of-new-malicious-files-detected-every-day-increases-by-52-to-360000-in-2020.
- [26] Khalilian, A., Nourazar, A., Vahidi-Asl, M., Haghighi, H., 2018. G3md: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families. *Expert Syst Appl* 112, 15–33. doi:10.1016/j.eswa.2018.06.012.
- [27] Kolter, J.Z., Maloof, M.A., 2004. Learning to detect malicious executables in the wild. *Association for Computing Machinery*, New York, NY, USA, pp. 470–478. doi:10.1145/1014052.1014105