



**Faculty of Information and Communication Technology**

**ENHANCEMENT OF STATIC CODE ANALYSIS MALWARE  
DETECTION FRAMEWORK FOR ANDROID CATEGORY-BASED  
APPLICATION**

**Azmi Bin Aminordin**

**Doctor of Philosophy**

**2021**

**ENHANCEMENT OF STATIC CODE ANALYSIS MALWARE DETECTION  
FRAMEWORK FOR ANDROID CATEGORY-BASED APPLICATION**

**AZMI BIN AMINORDIN**

**A thesis submitted  
in fulfilment of the requirements for the degree of Doctor of Philosophy**

**Faculty of Information and Communication Technology**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2021**

## **DECLARATION**

I declare this thesis entitled “Enhancement of Static Code Analysis Malware Detection Framework for Android Category-Based Application” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : .....

Name : .....

Date : .....

## APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Doctor of Philosophy.

Signature : .....

Supervisor Name : .....

Date : .....

## **DEDICATION**

This thesis is dedicated to Almarhumah Zainaf Bte Maidin, family, supervisors and all my friends which without whom none of my success would be possible.

## ABSTRACT

Android has become the number one mobile operating system in term of worldwide market share since May 2012. The highest demand and the open source factors had brought Android operating system into main target of malware creator. Two approaches introduced to detect malware in Android mobile environment namely static analysis and dynamic analysis. Static analysis is where the static features are examined. Too many features used, features extraction time consuming and the reliability of accuracy result by various machine learning algorithm are the main issues spotted in static analysis approach. As such, this thesis investigates the whole Android static analysis framework in detecting and classifying mobile malware. The early study found that two static features that are often used (permission and API calls) with the right mapping are sufficient to analyse the Android malware. The new permission(s) toward API call(s) mapping for Android level 16 to 24 is constructed based on Android official developer guideline references where previously these two features are mapped without using the standard guideline. On experimenting and analysing the framework, there are 4767 benign applications from 10 different categories was collected from Android official market place and 3443 malware applications was collected from AndroZoo dataset. All benign files are then scanned through VirusTotal to ensure that all collected files are free from virus. On extracting the desired features, a new automation of feature extraction using Depth First Search (DFS) with sequential search are introduced and succeed to extract the targeted features with consideration of no limitation on application file size also no limitation on file number. In order to enables machine learning to train faster and reduces the complexity of a machine learning model, the information gain features selection is applied towards the extracted features. Four types of machine learning algorithm were tested with four different kind of splitting dataset techniques separately. The result shows that the detection of malware within application category achieves higher accuracy compared to application with non-category based. In increasing the reliability, the results obtained are then validated by using statistical analysis procedure which each machine learning classification algorithm are iterate 50 times. The validation results show that Random Forest with 10-folds cross validation spitting dataset achieved 8 highest performance compared to benchmark study and two other classifiers. This study suggests the work to combine the optimization of feature selection and algorithm parameters to achieve higher accuracy and acquire more reliable comparison.

## **PENINGKATAN RANGKA KERJA PENGESANAN PERISIAN ANALISA KOD STATIK BAGI APLIKASI ANDROID BERASASKAN KATEGORI**

### **ABSTRAK**

*Semenjak Mei 2012, Android telah menjadi sistem pengoperasian mudah alih nombor satu dari sudut pasaran di serata dunia. Faktor permintaan yang tinggi dan konsep keterbukaan sumber telah mendorong sistem pengoperasian Android menjadi sasaran utama oleh pembangun perisian hasad. Terdapat dua pendekatan yang diperkenalkan untuk mengesan perisian hasad dalam persekitaran mudah alih Android iaitu secara analisa statik dan analisa dinamik. Analisis statik adalah di mana ciri statik dikaji. Terlalu banyak ciri yang digunakan, pengekstrakan fitur memakan masa dan kebolehpercayaan ketepatan yang dihasilkan oleh pelbagai algoritma pembelajaran mesin adalah masalah utama yang dilihat dalam pendekatan analisis statik. Tesis ini menyiasat rangka kerja bagi mengesan perisian hasad di dalam perisian mudah alih Android menerusi kaedah analisa kod statik dan mencadangkan penambahbaikan terhadap rangka kerja analisa statik bagi perisian Android. Kebenaran yang memetakan panggilan API untuk Android tahap 16 hingga 24 telah dibangunkan berdasarkan piawai daripada pembangun rasmi Android di mana sebelum ini kedua-dua fitur tersebut dipetakan tanpa menggunakan garis panduan piawai. Selanjutnya, proses dan algoritma baru bagi mengekstrak kebenaran dan panggilan API untuk mempercepat proses pengekstrakan dibangunkan. Di dalam pelaksanaan eksperimen dan penganalisan rangka kerja, sebanyak 4767 aplikasi bersih dari 10 kategori yang berbeza dikumpulkan dari pasaran rasmi Google dan 3443 aplikasi perisian hasad dikumpulkan dari dataset AndroZoo. Semua fail bersih kemudian diimbas melalui VirusTotal untuk memastikan semua fail yang dikumpulkan bebas dari virus. Automasi pengekstrakan fitur yang diperkenalkan berjaya mengekstrak fitur-fitur yang disasarkan dengan tidak mempunyai had pada saiz fail aplikasi juga tidak mempunyai had pada bilangan fail. Empat jenis algoritma pembelajaran mesin telah diuji dengan empat jenis teknik dataset secara berasingan. Hasil kajian mendapati, pengesanan perisian hasad berasaskan kategori mencapai ketepatan yang lebih tinggi berbanding aplikasi yang tidak berasaskan kategori. Hasil yang diperolehi kemudiannya disahkan melalui prosedur analisis statistik di mana setiap algoritma pengelasan pembelajaran mesin akan berulang sebanyak 50 kali. Hasil pengesanan menunjukkan bahawa pembelajaran mesin secara Random Forest dengan 10 kali liputan pengesanan silang mengumpul sebanyak 8 prestasi tertinggi dibandingkan dengan kajian tanda aras dan tiga pengelas lain. Tesis ini menyarankan kerja lanjutan terhadap kombinasi pengoptimuman pemilihan fitur dan parameter algoritma dilaksanakan bagi mencapai ketepatan yang lebih tinggi serta memperoleh perbandingan yang lebih baik.*

## ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful, for without Him, I would never be where I am today. Not to forget, greetings and eulogy to our beloved, The Great Prophet Muhammad SAW, for without him, we'll still be living the age of darkness.

I am deeply indebted to lecturers of UTeM, especially those in FTMK, and to my supervisors: Associate Professor Dr. Hj. Mohd Faizal Abdollah and Associate Professor Dr. Hjh. Robiah Yusof for their helps, supports, interests, valuable hints, guides and patience.

I am truly and forever obliged to Syarimasni Shamsuddin, for without her endearing support and loving company, I wouldn't make this far, and with my kids that I have the greatest blessing from Him and the joy of my life. Especially, I would like to forever give my deepest and most special thanks to my parents, my parents-in-law, my siblings, whose life-long teaching enabled me to complete this work and have pushed me to go this far. Special thanks to all my colleagues at UiTM Melaka and my friends who's always supported in many ways for completing this work.



## TABLE OF CONTENTS

	<b>PAGE</b>
<b>DECLARATION</b>	
<b>APPROVAL</b>	
<b>DEDICATION</b>	
<b>ABSTRACT</b>	<b>i</b>
<b>ABSTRAK</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>TABLE OF CONTENTS</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF APPENDICES</b>	<b>xiii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiv</b>
<b>LIST OF PUBLICATIONS</b>	<b>xvii</b>
<b>CHAPTER</b>	
<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Background	1
1.2 Problem statement	6
1.3 Research questions	7
1.4 Objectives	8
1.5 Scope of study	8
1.6 Research significance	9
1.7 Thesis organization	10
1.8 Summary	11
<b>2. LITERATURE REVIEW</b>	<b>12</b>
2.1 Introduction	12
2.2 Android operating system	12
2.2.1 Android history	13
2.2.2 Android architecture	13
2.2.3 Android application	16
2.2.4 Android security	19
2.2.5 Android permission model	21
2.2.6 Android API	25
2.2.7 Sensitive API	26
2.3 Mobile malware	26
2.3.1 Malware definition	27
2.3.2 Mobile malware taxonomy	28
2.3.3 Mobile malware on the rise	34
2.4 Android Malware Detection (AMD) system	37
2.4.1 Android malware detection phase	38
2.4.2 Android malware analysis techniques	41
2.4.3 Android malware identification techniques	43
2.4.3.1 Signature based detection	43
2.4.3.2 Anomaly-based	44
2.4.3.3 Specification-based	45

2.4.3.4	Hybrid-based	45
2.4.4	AMD platform	45
2.5	Previous studies of mobile malware analysis techniques	46
2.5.1	Static analysis	46
2.5.2	Dynamic analysis	53
2.6	Feature extraction and selection	56
2.6.1	Feature extraction for Android malware detection	56
2.6.2	Feature selection methods	57
2.7	Static analysis framework issues and key focus area	62
2.7.1	Dataset issues	63
2.7.2	Pre-processing	65
2.7.2.1	Mapping of API calls with permission	66
2.7.2.2	Feature extraction tools and techniques	67
2.7.2.3	Binary feature vector	69
2.7.3	Application category	70
2.7.4	Validation phase	72
2.7.5	Machine learning classifier	72
2.8	Performance validation	82
2.8.1	Majority criterion on selecting dataset splitting	83
2.8.2	Classification accuracy on machine learning algorithm	84
2.8.3	Statistical validation on classification accuracy	84
2.9	Implication and gaps	91
2.10	Summary	94
<b>3.</b>	<b>RESEARCH METHODOLOGY</b>	<b>95</b>
3.1	Introduction	95
3.2	Research design	96
3.2.1	Exploration phase	97
3.2.2	Implementation phase	98
3.2.2.1	Research development process	98
3.2.2.2	Performance measurement process	99
3.2.3	Validation process	100
3.3	Detailed research development process	101
3.3.1	Dataset preparation	102
3.3.2	Features used	105
3.3.3	Feature mapping	106
3.3.4	Feature extraction	107
3.3.5	Feature selection	107
3.3.6	Binary feature vector	108
3.3.7	Classification	108
3.4	Development tools and environment setup	110
3.5	Summary	111
<b>4.</b>	<b>PREPROCESSING RESULTS</b>	<b>113</b>
4.1	Introduction	113
4.2	Mapping of API calls toward permissions	113
4.3	Automation of feature extraction	119
4.3.1	MAPE flowchart	127
4.4	Generation of binary feature vector	130

4.5	Information gain filter method	134
4.6	Summary	136
<b>5.</b>	<b>RESULT AND DISCUSSION</b>	<b>137</b>
5.1	Introduction	137
5.2	Implementation of machine learning classifiers with different splitting dataset	137
5.3	Results and discussions: comparative of machine learning accuracy	139
5.4	Results and discussions: comparative of splitting dataset technique	144
5.5	Results and discussions: comparison with the benchmark study	151
5.6	Issues of machine learning classification result	153
5.7	Performance validation on splitting dataset technique	156
5.8	Data preparation for performance validation	157
5.9	Statistical validation of proposed framework	158
5.10	Normality test	159
5.11	Performance comparison	165
5.12	Summary	173
<b>6.</b>	<b>CONCLUSION</b>	<b>174</b>
6.1	Introduction	174
6.2	Research findings	174
6.3	Research contributions	176
6.4	Limitation of the study	177
6.5	Recommendation of future works	178
6.6	Summary	179
	<b>REFERENCES</b>	<b>181</b>
	<b>APPENDICES</b>	<b>221</b>

## LIST OF TABLES

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
2.1	Files and folders inside APK file (Developer, 2010)	17
2.2	Limitation of Android application-market ecosystems	20
2.3	Types of protection level (Developer, 2009)	24
2.4	Mobile malware attacking sources	36
2.5	Comparison of static features used by previous studies	49
2.6	Android malware dataset	64
2.7	Previous existing techniques for API extraction	68
2.8	Approach to generate binary feature vector	69
2.9	Previous study on Android category-based malware detection	72
2.10	Classifier used in previous experiments	73
3.1	A summary of the investigation phase	97
3.2	List of evaluation	100
3.3	List of sub-category name	105
3.4	Summary of classifier parameters	109
4.1	Number of API calls toward permission found by API level	119
4.2	Processing time results by MAPE	129
4.3	Top 20 features frequency in books and references and communication category	133
5.1	Accuracy results	140

5.2	Average percentage score by each machine learning algorithm	143
5.3	Score of splitting dataset using decision tree J48 algorithm	145
5.4	Score of splitting dataset using Naïve Bayes algorithm	147
5.5	Score of splitting dataset using random forest algorithm	146
5.6	Score of splitting dataset using SVM algorithm	147
5.7	Split technique scores by machine learning algorithm	147
5.8	Summation of total split scores percentage	150
5.9	Performance comparison with benchmark study	152
5.10	Top 3 higher accuracies obtained by machine learning algorithm	154
5.11	Mean value of 4 splitting dataset techniques	157
5.12	Test of normality for decision tree J48 classification accuracy of no-category (NoCate) based and category-based apps	160
5.13	Test of normality for Naïve Bayes classification accuracy of no-category (NoCate) based and category-based apps	161
5.14	Test of normality for random forest classification accuracy of no-category (NoCate) based and category-based apps	161
5.15	Test of normality for SVM classification accuracy of no-category based and category-based apps	162
5.16	Ranks for classification accuracy of category-based detection	163
5.17	Statistic test results using Mann-Whitney <i>U</i> test for classification accuracy of category-based against no-category based features	164
5.18	Average of accuracy obtained by category on different classifier	166
5.19	Score comparison of average accuracy on category-based against no-category-based	172

## LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	A general framework of Android malware static analysis (Rana and Sung, 2018)	2
1.2	Thesis organization	11
2.1	Android software stack (Source: Developer, 2011)	14
2.2	Percentage of infected customer's smartphone (McAfee Lab 2018)	27
2.3	General mobile malware taxonomy	29
2.4	Taxonomy of Android malware detection technique	37
2.5	Distribution of static feature used in Android malware detection	52
2.6	General framework of static analysis malware detection (M. Rana & A. Sung, 2018)	62
2.7	Example of separating multiple lines	74
2.8	Optimal SVM hyperplane	75
2.9	Decision tree	77
2.10	Random forest	78
2.11	A summary of performance measurement processes	91
2.12	New processes for enhancement framework	93
3.1	Research design	96

3.2	Performance measurement and validation	101
3.3	Research development process	102
3.4	Summary of dataset preparation phase	104
3.5	Summary of feature type	106
3.6	MS Excel snapshot of binary feature vector	108
4.1	Architecture of API call toward permission mapping	114
4.2	Web page snapshot of API level 16 package names	115
4.3	Web page snapshot of class names in "android.account" package on API level 16	116
4.4	Web page snapshot of "AccountManager" class information	117
4.5	Web page snapshot of permission keyword highlighted	118
4.6	Depth first search	121
4.7	New algorithm Multiple Android Package Extractor (MAPE) using DFS with sequential search	121
4.8	Code snippet of MAPE program	122
4.9	Existing architecture of extracting API calls	123
4.10	Propose architecture (MAPE)	124
4.11	MAPE interface (Step 1)	124
4.12	MAPE interface (Upload Successfully)	125
4.13	MAPE interface (Step 2: Disassemble)	125
4.14	Snapshot of results by MAPE	126
4.15	MAPE flowchart	128
4.16	Sequential search pseudo code	130
4.17	Architecture of the automation binary feature vector	130

4.18	Binary feature generation flowchart	131
4.19	Result produces by binary feature vector generator	132
4.20	Example of education.csv file	133
4.21	General framework of supervised feature selection	135
5.1	Percentage score by 4 classifiers on 10-folds cross validation	141
5.2	Percentage score by 4 classifiers on 70% training-30% testing	141
5.3	Percentage score by 4 classifiers on 80% training – 20% testing	142
5.4	Percentage score by 4 classifiers on 90% training – 10% testing	142
5.5	Overall percentage of score by all classifiers on 4 types of splitting technique	144
5.6	Splitting technique percentage scores on decision tree J48 algorithm	148
5.7	Splitting technique percentage scores on Naïve Bayes algorithm	148
5.8	Splitting technique percentage scores on random forest algorithm	149
5.9	Splitting technique percentage scores on SVM algorithm	149
5.10	Overall percentage scores for all splitting technique	150
5.11	Proposed statistical validation process for Android application category-based malware detection	159
5.12	Average of classification accuracies percentage of no-category (NoCate) and book and references (BookRef) category using 10 folds cross validation	167
5.13	Average of classification accuracies percentage of no-category (NoCate) and communication (Comm) category using 10 folds cross validation	167



5.14	Average of classification accuracies percentage of no-category (NoCate) and education (Edu) category using 10 folds cross validation	168
5.15	Average of classification accuracies percentage of no-category (NoCate) and entertainment (Enter) category using 10 folds cross validation	168
5.16	Average of classification accuracies percentage of no-category (NoCate) and health category using 10 folds cross validation	169
5.17	Average of classification accuracies percentage of no-category (NoCate) and lifestyle (LStyle) category using 10 folds cross validation	169
5.18	Average of classification accuracies percentage of no-category (NoCate) and music and audio (MscAud) category using 10 folds cross validation	170
5.19	Average of classification accuracies percentage of no-category (NoCate) and personalization (Person) category using 10 folds cross validation	170
5.20	Average of classification accuracies percentage of no-category (NoCate) and photography (Photo) category using 10 folds cross validation	171
5.21	Average of classification accuracies percentage of no-category (NoCate) and tools category using 10 folds cross validation	171

## LIST OF APPENDICES

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	Apart of API calls toward permission list on API level 24	221
B	Features code and feature name	224
C	Top 20 feature frequencies by category	236

## LIST OF ABBREVIATIONS

AAPT	-	Android Assets Packing Tool
AMD	-	Android malware detection
APK	-	Android Package
arff	-	Attribute-relation file format
ART	-	Android Runtime
AOSP	-	Android Open Source Project
AOT	-	Ahead of Time
API	-	Application Programming Interface
CEO	-	Chief Executive Officer
CFS	-	Correlation Feature Selection
CSV	-	Comma Separated Values
DDoS	-	Distributed Denial of Service
DEX	-	Dalvik Executable
DFS	-	Depth first search
DT	-	Decision Tree
DVM	-	Dalvik Virtual Machine
FPR	-	False Positive Rate
GA	-	Genetic Algorithm
GeFS	-	Genetic Features Selection
ID	-	Identification

IDS	-	Intrusion Detection System
IG	-	Information Gain
IMEI	-	International Mobile Equipment Identity
IMSI	-	International Mobile Subscriber Identity
IRC	-	Internet Relay Chat
JNI	-	Java Native Interface
JVM	-	Java virtual machine
KPCA	-	Kernel Principal Component Analysis
KNN	-	K-Nearest Neighbour
MAPE	-	Multiple Android Package Extractor
MIB	-	Management Information Base
MLP	-	Multi-Layer Perceptron
MMS	-	Multimedia Messaging Service
OHA	-	Open Handset Alliance
OS	-	Operating system
PC	-	Personal Computer
PIN	-	Personal Identification Number
RPC	-	Remote Procedure Call
SDK	-	Software Development Kit
SMS	-	Short Message Service
SMO	-	Sequential Minimal Optimization
SPSS	-	Statistical Product and Service Solutions
SVM	-	Support vector machine
TF-IDF	-	Term Frequency Inverse Document Frequency
TPR	-	True Positive Rate

- VBA - Visual Basic for Application
- VM - Virtual machine
- WEKA - Waikato Environment for Knowledge Analysis

## LIST OF PUBLICATIONS

Aminordin, A., M.A., F. And Yusof, R., 2018. Android malware classification base on application category using static code analysis. *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 20, pp.6853-6863.

Aminordin, A., Faizal, M.A., Robiah, Y., Mukhlis, A. and Arif, F., 2018. Multiple android package files extractor in mining request permissions and API calls. *Journal of Advanced Manufacturing Technology (JAMT)*, vol. 12 (2), pp.11-24.

Aminordin, A., Abdollah, M.F., Yusof, R. and Ahmad, R., 2017. Preliminary findings: revising developer guideline using word frequency for identifying apps miscategorization. *Proceedings of the Second International Conference on the Future of ASEAN (ICoFA), Perlis Malaysia*, vol. 2, pp.123-131.

# CHAPTER 1

## INTRODUCTION

### Background

In year 2019, Kaspersky lab detected about 3.4 million malicious installation packages (Chebyshev, 2020) which was nearly 1.5 times fewer than the previous year. Despite a decrease in the number of detected malicious installation packages, the number of attacks on personal data grew from 40,386 in 2016 to 67,500 in 2019.

Furthermore, in second quarter of 2019 a steady rise in the number of mobile threat detected (Chebyshev et al., 2020). Even though Android Open Source Project (AOSP) is committed to secure Android smartphone OS, it is also susceptible to the social-engineering attacks (Faruki et al., 2015). Every year, Android malware threat report is published by several anti-virus companies for public access, yet, the smartphone users awareness on its potential risk is still at the lower rate (Qiao et al., 2016). Numerous attack vectors occur which compromise smartphone security (Samani and Beek, 2018). At present, Android is holding a global market share of 74.44%, hence, Android OS devices attack phenomena cannot be overstated. Information stealing and monetisation are two serious threats toward Android based smartphones which lead to financial charges for Android users.

Most mobile phone users think that their phones are safe and assume they can proceed to do the tasks that they prefer without putting them in any risk. A study by mobile application security firm (Reed, 2019) found that 70% out of 250 applications collected from Android official market place suffer from vulnerabilities that could lead to privacy leakage. Even though Google introduces permission-based guard, in which the mechanism is to

restrict access of the third-party Android applications to critical resources on Android devices through their Bouncer security in 2012, the mobile security firm still asserts that millions of Android users are facing high-risk vulnerabilities. This is because, not only sensitive information can be spread illegally, but it also could give negative impact to the economy. Hence, the existing technique of malware detection is continuously being explored and revised.

A detection of malicious activities in Android mobile phone is a crucial analysis process where it can be divided into two main approaches namely static analysis, and dynamic analysis. Static analysis detection refers to examining potential security problem program without performing or running the program. In other words, the inspection is done through each programming line and text used in various files. Figure 1.1 shows a general framework in detecting Android malware which consists of four main phases namely data collection, features extraction phase, features selection phase and classification phase.

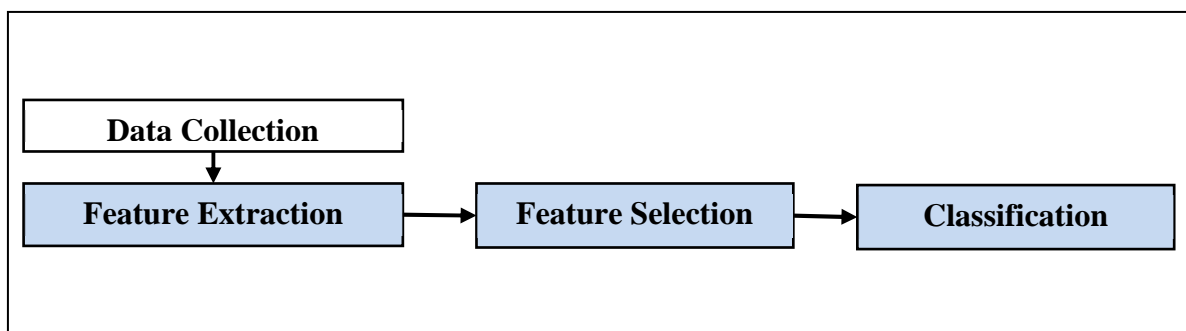


Figure 1.1: General framework of static analysis malware detection (Rana and Sung, 2018)

Static analysis is the most preferred method by many researchers (Bakour et al., 2018) because of its low computational time, ease of implementation and effectiveness in detecting malicious application. In the first phase, the dataset is prepared and processed to be in a suitable format to extract the desired features that will be used to structure the pattern that