

FEATURE SELECTION TO ENHANCE ANDROID MALWARE DETECTION
USING MODIFIED TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY
(MTF-IDF)

NURUL HIDAYAH BINTI MAZLAN

A thesis submitted in
fulfillment of the requirement for the award of the
Degree of Master of Information Technology

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

FEBRUARY 2019

ACKNOWLEDGEMENT

I would like to express my thanks to all that was involved in the process of conducting the research and writing of this thesis. The completion of this thesis could not have been possible without assistance numerous of people and parties. Firstly, I would like to express the deepest appreciation to my supervisor, Ts. Dr. Isredza Rahmi Binti A. Hamid, for her understanding, encouragement and guidance during my research study. Without her incredible patience during supervised my project, I would not have been able to finish my research. I am grateful to all lecturers of Faculty Computer Science and Information Technology that helping me with my coursework and academic research in the past few years. I also would like to thank my fellow postgraduate members that always support each other, and other parties who helped me throughout exploration in research. Appreciation also goes to everyone involved directly or indirectly towards the compilation of this thesis. Finally, special thanks to my parents, my brother and other family members for their continuous encouragement. This thesis would not have been able to complete without their continuous love and support.

ABSTRACT

This research synthesizes an evaluation of feature selection algorithm by utilizing Term Frequency-Inverse Document Frequency (TF-IDF) as the main algorithm in Android malware detection. The TF-IDF algorithm is used to filter Android features filtered before detection process. However, IDF is unaware to the training class labels and gives incorrect weight value to some features. Therefore, the proposed approach that is Modified Term Frequency – Inverse Document Frequency (MTF-IDF) algorithm give more focus on both sample and features to give correct weight value to some features. The proposed algorithm considered features based on its level of importance where weight given based on number of features involved in the sample. The related best features in the sample are selected using weight and priority ranking process using K-means. This ensures that only important malware features are selected in the Android application sample. These experiments are conducted on a sample collected from DREBIN. Comparison between existing TF-IDF algorithm and MTF-IDF algorithm have been made under various conditions such as tested on different number of sample size, different number of features used and integration of different types of features. The results showed that feature selection using MTF-IDF can improve Android malware detection analysis. It was proven that MTF-IDF is an effective Android malware detection algorithm regardless of different kinds of features or sample sizes used. MTF-IDF algorithm also proved that it can give appropriate scaling for all features in analyzing Android malware detection.

ABSTRAK

Penyelidikan ini mensintesis penilaian algoritma pemilihan ciri dengan menggunakan *Frequency-Inverse Frequency Document* (TF-IDF) sebagai algoritma utama dalam pengesanan malware Android. Algoritma TF-IDF digunakan untuk menapis ciri-ciri Android yang ditapis sebelum proses pengesanan. Walau bagaimanapun, IDF tidak menyedari label kelas latihan dan memberikan nilai berat yang salah kepada beberapa ciri-ciri Android. Oleh itu, pendekatan yang dicadangkan iaitu algoritma *Modified Frequency Inverse-Docment Frequency* (MTF-IDF) memberi tumpuan lebih kepada kedua-dua sampel dan ciri-ciri Android untuk memberikan nilai berat yang betul kepada beberapa ciri-ciri Android. Algoritma yang dicadangkan ini memilih ciri-ciri Android berdasarkan tahap kepentingannya dimana berat diberikan berdasarkan bilangan ciri-ciri Android yang terlibat dalam sampel. Ciri-ciri Android terbaik yang berkaitan dalam sampel dipilih menggunakan berat dan proses ranking keutamaan menggunakan K-means. Ini memastikan bahawa hanya ciri-ciri Android malware yang penting dipilih dalam sampel aplikasi Android. Eksperimen ini dijalankan pada sampel yang dikumpulkan daripada sampel data DREBIN. Perbandingan antara algoritma TF-IDF sedia ada dan algoritma MTF-IDF telah dibuat dibawah pelbagai syarat seperti diuji pada bilangan sampel yang berlainan, menggunakan bilangan ciri-ciri Android yang berbeza dan integrasi pelbagai jenis ciri-ciri Android. Hasilnya menunjukkan bahawa pemilihan ciri-ciri Android menggunakan MTF-IDF boleh meningkatkan analisis pengesanan malware Android. Hal ini terbukti bahawa MTF-IDF adalah algoritma pengesanan malware Android yang berkesan tanpa mengambil kira pelbagai jenis ciri-ciri Android atau saiz sampel data yang digunakan. Algoritma MTF-IDF juga membuktikan bahawa ia dapat memberikan skala yang sesuai untuk semua ciri dalam menganalisis pengesanan malware Android.

TABLE OF CONTENTS

DECLARATION	ii
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
ABSTRAK	v
TABLE OF CONTENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	xi
LIST OF ALGORITHMS	xii
LIST OF ABBREVIATIONS	xiii
LIST OF APPENDIX	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Background Overview	1
1.2 Research Motivation	2
1.3 Problem Statement	3
1.4 Objectives of the Research	5
1.5 Scope of Research	5
1.6 Significance of Research	6
1.7 Structure of the Thesis	6
CHAPTER 2 LITERATURE REVIEW	8
2.1 Introduction	8

2.2	Android Malware	8
2.3	Android Architecture	9
2.4	History of Android Malware Detection Approach	12
2.5	Android Malware Feature	13
2.5.1	Static Features	13
2.5.2	Dynamic Features	14
2.5.3	Hybrid Features	15
2.5.4	Machine Learning	16
2.5.5	Comparison of Works in Android Malware Detection	17
2.6	Feature Selection Approach	18
2.6.1	Term Frequency-Inverse Document Frequency (TF-IDF)	18
2.7	Chapter Summary	19
CHAPTER 3 RESEARCH METHODOLOGY		21
3.1	Introduction	21
3.2	Android Malware Detection Framework	21
3.2.1	Preprocessing	23
3.2.2	Data Manipulation	25
3.2.3	Malware Analysis and Detection	29
3.2.4	Constructing Feature Matrix	32
3.2.5	Evaluation of Feature Selection for Android Malware Detection	33
3.3	Performance Metric	35
3.4	Chapter Summary	35
CHAPTER 4 RESULTS AND DISCUSSION		37
4.1	Introduction	37
4.2	Feature Selection for Android Malware Detection	37
4.3	Comparison of Weighted Based Feature Selection Technique with Existing Algorithm	39
4.3.1	Different Number of Sample Size	39

4.3.2	Different Number of Features Used	44
4.3.3	Integration Different Types of Feature	47
4.4	Chapter Summary	52
CHAPTER 5 CONCLUSION AND FUTURE WORK		53
5.1	Introduction	53
5.2	Contribution of the Research	54
5.3	Recommendations for Future Work	55
5.4	Concluding Remarks	56
REFERENCES		57
APPENDIX		63
VITAE		68



LIST OF TABLES

1.1	Comparison of devices types worldwide market share	1
2.1	Comparison of related works	17
3.1	Parts of dangerous permission features	24
3.2	Parts of API call features	24
3.3	The difference between TF-IDF algorithm and MTF-IDF algorithm	27
3.4	Example of set of Android features data	27
3.6	Example of set of sample	30
3.7	Centroid for two groups of cluster	31
3.8	Centroid value for each sample	31
3.9	Centroid for two groups of cluster	31
3.10	Centroids for two groups of cluster	32
3.11	Finalized centroid for all samples	32
4.1	Android malware detection experimental setup	39
4.2	True positive (TP) and false positive (FP) rates on different sample size for API call based feature using Random Forest algorithm	42
4.3	True positive (TP) and false positive (FP) rates on different sample size for dangerous permission based feature using Random Forest algorithm	43
4.4	True positive (TP) and false positive (FP) rates on API call based feature tested on Random Forest algorithm	46

4.5	True positive (TP) and false positive (FP) rates on dangerous permission based feature tested on Random Forest algorithm	47
4.6	List of features used	48
4.7	Performance comparison of accuracy based on algorithm using combination of API call and dangerous permission	51



LIST OF FIGURES

2.1	Android Architecture	10
3.1	Android malware feature selection approach	22
4.1	Number of detected malware over number of sample for API call based feature	40
4.2	Number of detected malware over number of samples for dangerous permission based feature	41
4.3	Accuracy value using different number of samples size for API call based feature tested on Random Forest algorithm	41
4.4	Accuracy value using different number of samples size for dangerous permission based feature tested on Random Forest algorithm	42
4.5	Number of malware detected using different number of features for API call based feature	44
4.6	Number of malware detected using different number of features for dangerous permission based feature	44
4.7	Accuracy value using different number of API call based feature tested on Random Forest	45
4.8	Accuracy value using different number of dangerous permission based feature tested on Random Forest	46
4.9	Number of malware detected on integrated features using Random Forest algorithm	50
4.10	Performance comparison of accuracy using combination of features using Random Forest	50

LIST OF ALGORITHMS

3.1	Android malware detection algorithm	23
3.2	Algorithm for weighted calculation	25
3.3	Algorithm for clustering the sample	30



LIST OF ABBREVIATIONS

API	-	Application Program Interface
DF	-	Document Frequency
FN	-	False Negative
FP	-	False Positive
GPS	-	Global Positioning System
GUI	-	Graphic User Interface
IDF	-	Inverse Document Frequency
MTF-IDF	-	Modified Term Frequency-Inverse Document Frequency
OHA	-	Open Handset Alliance
SVM	-	Support Vector Machine
TF	-	Term Frequency
TF-IDF	-	Term Frequency-Inverse Document Frequency
TP	-	True Positive
URI	-	Uniform Resource Identifier
URL	-	Uniform Resource Locator
OHA	-	Open Handset Alliance
WEKA	-	Wakaito Environment for Knowledge Analysis

LIST OF APPENDIX

APPENDIX	TITLE	PAGE
A	Table A.1: Gantt chart of research activities	63
B	Table B.1: Application Program Interface (API) call based feature	64
C	Table B.2: Dangerous permission based feature	67



PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

CHAPTER 1

INTRODUCTION

1.1 Background Overview

Web browsing and checking e-mails are among tasks that can be performed using mobile devices. January 2018 Global Digital Report (We Are Social, 2019) said 61% of population in Malaysia believe that new technologies offer more opportunities than risks, while 60% prefer to complete tasks digitally whenever possible. Aside from that, mobile devices are able to efficiently synchronize data between multiple devices using cloud-based services.

These portable and connected devices encourage users to use applications to perform everyday tasks (CyberSecurity Malaysia, 2017). CyberSecurity also reported the decline sale of personal computers starting early 2010. Meanwhile, market share worldwide shows that mobile devices have been selected as the most devices used from December 2016 until 2018. Table 1.1 shows comparison of devices types worldwide market share (Statista, 2019). Mobile devices shows high percentage with 50.31% compare to personal computer, 44.79%. The number of smartphone in 2017 increased 53.99%, while personal computer decreased to 45.68%. In 2018, the number of smartphone used decrease to 52.95% but still higher than personal computer with 52.07%. This shows that mobile devices are more preferred by users compared to traditional personal computer.

Table 1.1: Comparison of devices types worldwide market share (Statista, 2019)

Device Type	2016	2017	2018
Personal Computer	44.79%	45.68%	52.07%
Mobile Devices	50.31%	53.99%	52.95%

The increase in number of mobile devices users encourage attackers to mine users' personal information such as names, contacts, bank account passwords, credit card numbers, usernames and passwords for online banking, or memorable and private pictures for their own profit. Other than stealing the said data, attackers could gain access for the purpose of damaging the device or bothering the users' privacy (Felt *et al.*, 2011). The attackers illegally took advantage of the vulnerability of a device by installing malicious applications and gain unauthorized remote access. Android platform has introduced security solutions that can hinder the installation of malware, for example Android permission system. Android application has to run request permission from users to perform certain tasks on Android devices, such as sending a message, during installation process. However, users sometimes tend to grant permission without realizing that the permissions they are providing are capable to weaken the protection provided by Android platform.

Due to the increasing number of attackers, Android malware researchers from all field come out with the idea to reduce cybercrime from affecting Android user, such as using weighted based algorithm. Several research such as Li *et al.* (2016) and Zhang *et al.* (2014) on detecting Android malware using weighted based algorithm by utilizing Term Frequency Inverse Document Frequency (TF-IDF) as the main algorithm in Android malware feature selection has been come out. Lee *et al.* (2014) used TF-IDF algorithm in calculating the weight each of the Android static features to each of the training malicious applications. Li *et al.*, (2016) applied TF-IDF algorithm to calculate the weight of every dangerous Application Program Interface (API) in the feature vector. Android sample has been collected from DREBIN (Arp *et al.*, 2014). Research that involved dangerous permission based feature and Application Program Interface (API) call based feature has been analyse to learn classifier from the sample to improve Android malware detection.

1.2 Research Motivation

Malware stands for 'malicious software' (Sami *et al.*, 2010). Malware is typically used to refer to any kind of software that can cause damage to a single computer, server or computer network. As the number of smartphones increases, the number of malware that targets popular mobile devices platform also increased (Abawajy *et al.*,

2017). Android based mobile devices have its own operation system with additional capabilities and capacities with functions similar to a computer. Aside from that, Android based mobile devices have combination of different network connectivity with high-speed data networking capabilities and geo-location services (Vidas *et al.*, 2011). Users can easily be forced to subscribe call or message, remote control of money transfer and extortion to ransomware. G Data (2016) reported the introduction of 440,000 new Android malware strains in the first quarter of 2015, which means that every 18 seconds, a new mobile malware strain for Android is discovered (GData, 2016). Thus, it is clear that mobile devices have become the main target for malware attackers to seek profit by mining users' confidential information.

Over the years, there has been an increase in the technology, diversity, and complexity of Android malware in response to increased user awareness and countermeasures. Android malware research may improve Android based security and improve mobile protection for users and organizations. This may accurately distinguish between normal application and malicious software.

There have been some research done to detect Android malware by using automatic dynamic malware analysis combined with machine learning algorithm (Mas'ud *et al.*, 2014). During analysis, large number of features may include irrelevant features and will cause false result in machine learning algorithm (Shabtai *et al.*, 2012). Using feature selection, Android malware detection can be run efficiently using machine learning algorithm (Mas'ud *et al.*, 2014). Other than that, selecting the most useful subset of features from huge number of Android features can remove noisy and irrelevant data from samples, leading to more accurate results (Feizollah *et al.*, 2015).

This research improves Android malware detection by using the Modified Term Frequency-Inverse Document Frequency (MTF-IDF) to select related best features in the sample using weight and priority ranking process. This increases the effect of important malware features selected in the Android application sample.

1.3 Problem Statement

Android malware detection process has drawn researchers' attention. Up till now, Android malwares have become more complex and attackers are able to avert the

Android malware detection techniques. Existing Android malware detection processes are based on static and dynamic analysis (Jang *et al.*, 2016). Static analysis is a fast and efficient method. However, static analysis approach is prone to high false negative rates due to evolution in code basis and code repacking. Dynamic analysis aims to provide effective and efficient methods to extract unique patterns of each malware family based on its behavior. In order to enhance malware detection, dynamic analysis requires more features such as system call and network activities.

Even though many research have works on Android malware detection using static and dynamic, number of Android malware attack keep increasing. This shows that this technique still cannot overcome this problem due to the wrong selection of features. Due to that issue, selection of features that can be used to detect malware is crucial in order to achieved accurate result with low false positive rate. Feature selection is one of the core concepts in machine learning which hugely impacts the performance of algorithm (Raheel Shaikh, 2018). The data features that have been used to train machine learning algorithms have a huge influence on the performance that could be achieved. Irrelevant or partially relevant features can negatively impact algorithm performance. Good selection of features can reduce noise and irrelevant features which lead to better malware detection result.

Thus, it is an urgent demand to identify Android malwares based on feature selection. In particular, the following two research issues have been address in this thesis:

1. *How to detect Android malware in an efficient manner:* Detection is a crucial aspect in fighting against Android malwares. There are so many Android malwares approach that have been proposed in order to solve detection problem. However, the number of Android malwares keeps increasing which shows that attackers are becoming more advanced and can easily bypass the Android malware detection techniques. Thus, a new solution for Android malware detection is required where new feature selection approach by modifying the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm is proposed.
2. *How to measure the feature selection approach:* In order to measure the effectiveness of the feature selection approach, the performance of the proposed MTF-IDF algorithm were evaluated based on accuracy value.

1.4 Objectives of the Research

To achieve the research aim, three main research objectives are identified and need to be fulfilled:

- (i) to propose feature selection technique using Term Frequency-Inverse Document Frequency (TF-IDF) for Android malware detection.
- (ii) to develop a modified TF-IDF (MTF-IDF) algorithm to select the best features that could improve Android malware detection accuracy.
- (iii) to evaluate the performance of the proposed MTF-IDF algorithm based on accuracy.

1.5 Scope of Research

The scope of the research is focusing on solving Android malware problem particularly in feature selection in detecting Android malwares. This research aim is to provide a solution that is highly accurate and efficient in identifying Android malwares. This research more focused on selected features instead number of sample, so only 1000 static and dynamic Android features samples are collected from DREBIN (Arp *et al.*, 2014). After preprocessing has been done, 17 types of Application Program Interface (API) call based which represents static feature and eight types of dangerous permission based features that represents dynamic feature were used in this research (Android Developers, 2018).

Then, these features are tested using TF-IDF and the proposed modified TF-IDF (MTF-IDF) algorithm. To classify unsupervised data into malware or benign, K-means clustering was used and applied based on rules to decide the class. Result collected runs performed using three types of machine learning algorithms; Random Forest, Decision Table and Adaboost. Those machine learning algorithm were selected because of all of it have capabilities in classified nominal class, binary class and missing class value. All machine learning algorithms used (Random Forest, Decision Table and Adaboost) the test option applied is cross-validation with 10-folds using WEKA default setting in WEKA tools. Finally, the performance of both algorithms was evaluated based on accuracy value performance.

1.6 Significance of Research

The TF-IDF algorithm is widely used in text classification research (Forman, 2008) because it is richer and more successful representation for retrieval and categorization purposes (Shabtai *et al.*, 2009). However, it is unaware to the training class labels and gives incorrect weight value to some features, which can lead to inappropriate scaling for some features (Forman, 2008). This research introduces an approach to detect Android malwares using Modified Term Frequency-Inverse Document Frequency (MTF-IDF). Android malwares are detected based on weight calculation from collection of Android samples. The MTF-IDF algorithm gives weight to the feature selection based on level of importance of the features in the set of sample. This cause the selection of features is more accurate and reliable than other approach. Comparison made between TF-IDF and MTF-IDF proved that the latter algorithm performed better than its predecessor.

1.7 Structure of the Thesis

This thesis consists of six main chapters and is organized as the following:

1. Chapter 1 describes introduction of the research including background overview, research motivation, statement of the problem, objectives of the research, scope of research, significance of research and structure of the research.
2. Chapter 2 describes a brief history of Android malware and overview on feature selection technique. This chapter also discusses about research related to Android malware detection approaches.
3. Chapter 3 describes research process and research framework.
4. Chapter 4 describes the development of feature selection using weighted based technique.
5. Chapter 5 describes performance evaluation metric of feature selection using weighted based technique.
6. Chapter 6 concludes the work and presents some recommendations for future works to improve research efficiency in detecting Android malwares.

This thesis was derived from the following publication:

1. Mazlan N.H., Hamid I.R.A. (2018). Using Weighted Based Feature Selection Technique for Android Malware Detection. International Conference on Mobile and Wireless Technology (pp. 54-64). Springer, Singapore. (Published)
2. Mazlan N.H., Hamid I.R.A. (2018). Evaluation of Feature Selection Algorithm for Android Malware Detection. International Journal of Engineering & Technology, 7(4.31), 311-315. (Published)



CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter discusses Android malware and the literature of algorithms involved during analyzing feature selection. This chapter also explains the differences between previous researches with this research. From the literature review, existing problems and the weaknesses of previous research can be overcome.

2.2 Android Malware

Most people today, like to use Android based mobile devices. Adults, teenagers and also kids have their own mobile devices because of their various capabilities for example accessing social networks and accessing online games. By using Android mobile devices, people can access many applications such as e-mails, transactions, maps, social networks and other features. Before this, computers or laptops are needed for sending e-mail, but now these devices are sufficient to do such task. Android mobile device is a software stack that includes an operating system, middleware and key applications (Saha, 2008). However, Android mobile devices are highly vulnerable to various types of attacks. Malware is a common way for launching attacks such as virus, spyware and Trojan (Coronado-de-alba *et al.*, 2013). Malware can affect private users, commercial companies and governments. Malware that successfully installed on mobile devices can result in the loss, unauthorized utilization or modification of large amounts of data and cause users to question the reliability of all the information on the network.

A comprehensive research of digital, social and mobile usage in Malaysia reported statistics from Digital Report Southeast Asia (We Are Social, 2019) said that the digital world experienced huge growth in 2018. Unique mobile users grew by 2%, up 376 thousand in 2018. Meanwhile, mobile social media users grew 2 million, a 10% increase in 2018. Digital Report for Southeast Asia also reported percentage of Internet in Malaysia users increase to 14%, which are 3 million new users compared to 2017. 24.08 million reported active as mobile internet users of total populations. This survey shows 59% mobile Internet user in this country may have more than one mobile device.

A mobile device has processing power and memory like a computer with additional capabilities such as different network connectivity with high-speed data networking and geo-location services. Those capabilities increase the number of mobile devices on the market. Therefore, the increasing number of modern mobile devices and communication infrastructure cause highly vulnerable to various types of attacks. Malware is a common way in launching attacks on mobile devices. Malware threats on mobile devices come in various forms such as viruses, Trojans, worms and mobile botnets (Suarez-Tangil *et al.*, 2014). Android is the most attractive prey because it has large user base and employs open source code. Malicious applications can be easily installed into the devices and be used for intrusive surveillance, platform exploit delivery, or other follow-on attacks. This phenomenon forces several security challenges such as leak of sensitive data, network or file system to be addressed.

2.3 Android Architecture

Android is the most famous open-source mobile platform (Darcey *et al.*, 2012). Nowadays, mobile users demand more functionality in customizing their devices. Figure 2.1 shows the Android Architecture that consist of layers which are application layer, framework layer, middleware layer and kernel layer (Men *et al.*, 2018). The uppermost layer which is application layer provides a set of core applications such as email, SMS, calendar, maps and contacts. All applications are written using the Java programming language. Framework layer is a software framework that is used to implement a standard structure of an application for a

specific operating system. Layer below framework layer, which is Android middleware layer, contains essential elements of Android as a mobile platform.

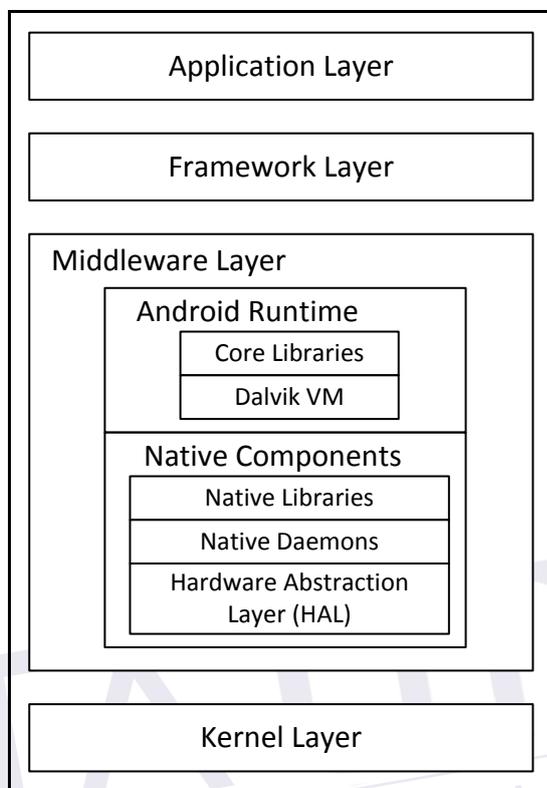


Figure 2.1: Android Architecture (Gandhewar *et al.*, 2011)

There are two parts in middleware layer, which are Android runtime system and native components. Core libraries in Android runtime provide most of the functionality available in the core libraries of the Java programming language. With Dalvik virtual machine (Dalvik VM) own instance, every Android application runs in its own process. The Dalvik VM executes files which are optimized for minimal memory footprint in the Dalvik Executable (.dex) format.

Native libraries and native daemons in the native components are written in C/C++. The native libraries greatly enrich the functionality and compatibility of Android platform for the development purpose, meanwhile the native daemons handle all interaction with the system in native level. Hardware Abstraction Layer (HAL) defines a standard interface to bridge the gap between hardware and software. Compared with the drivers located in the kernel layer, Android HAL holds most of

the hardware vendor specific implementation, for example, the APIs of audio device and camera.

The bottom layer is Linux Kernel which provides the basic functionalities of operating systems such as kernel drivers, power management and file system. Android basically relies on Linux for core system services such as security, process management, memory management, network stack and driver model. Between the hardware and the rest of the software stack, kernel acts as an abstraction layer.

Android has emerged as an innovative and open platform to fulfill the growing needs of the mobile marketplace. Google has taken a broad approach to examine the wireless community. The Open Handset Alliance (OHA) was formed in November 2007 to build better mobile phones. OHA consists of the most successful mobile companies including chip makers, handset manufactures, software developers and service providers. Google acquired Android Inc. in 2005, which then worked together with OHA members to develop a nonproprietary open standard platform based on technology developed at Android Inc. This technology keeps on improving. Although it has not made a huge dent in the tablet market yet, Google's Android operating system has been the top-selling smartphone platform for a while. That also means it attracts the interest of scammers and malware developers to steal data and money from unsuspecting Android users.

However, an attacker would get the most reliable, long term access to a victim's device by tricking them into installing a malicious application. The malicious application actually incorporates WebView and exploits the bug originated in Chromium, the open-source project that underlies Chrome and many other browsers (WIRED, 2019). Attackers could also use the bug to gain inappropriate device access by tricking users into clicking a malicious link that would then open through Android's Instant App feature. This component allows users to run a version of an application immediately without actually installing it. In that scenario, an attacker wouldn't have permanent, persistent access, but would have a limited window of time to start hoovering up a user's data or information about their mobile accounts. Either way, methods are quiet and inconspicuous compromises.

Google has worked to improve its ability to push patches across devices and minimize difficulties caused by variations in manufacturer implementation. However, there's still a very long way to go because of Android's ubiquity in all

4.3.2 Different Number of Features Used

Performance comparison of malware detected using different number of features for API call is shown in Figure 4.5. Number of malware detected using 10 API call based features is only 1 malware for TF-IDF and 15 malwares for MTF-IDF algorithm. Number of malware detected for both experiments increased with the increase in number of features used. The increase in number of malware detected between 10 features and 100 features is 342 for TF-IDF and 632 for MTF-IDF.

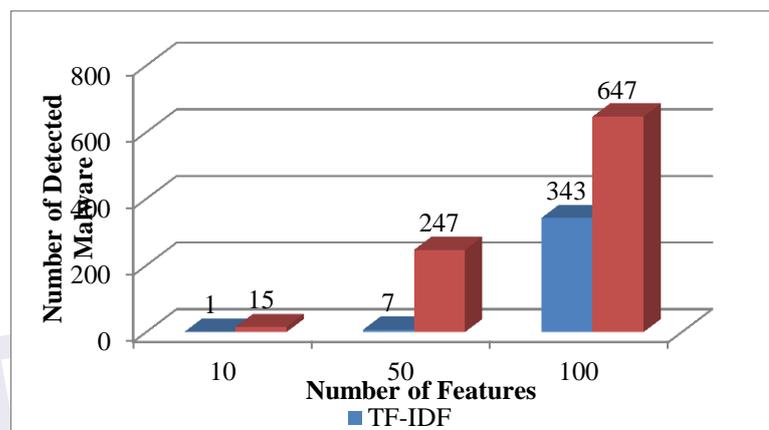


Figure 4.5: Number of malware detected using different number of features for API call based feature

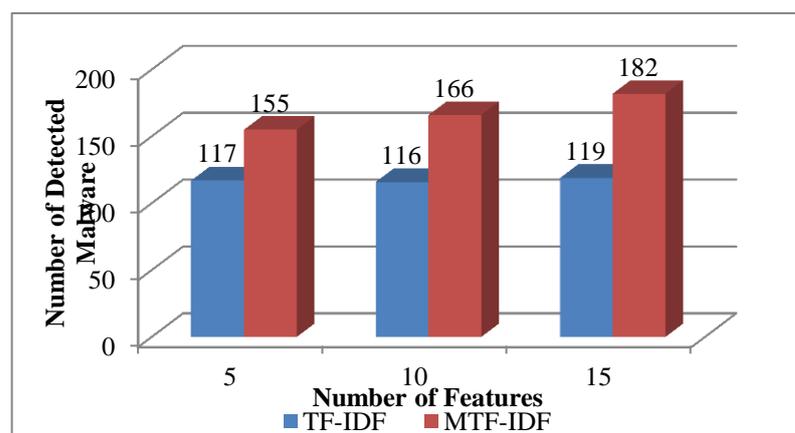


Figure 4.6: Number of malware detected using different number of features for dangerous permission based feature

Meanwhile, for dangerous permission based feature, number of malwares detected for both experiments decreased with the increase in number of features as in Figure 4.6. For TF-IDF algorithm, the different number of malware detected between 5 features and 15 features in the set of sample is 2 malwares, and 27 malwares for MTF-IDF algorithm.

Figure 4.7 shows the accuracy value based on different number of features for API call based feature. MTF-IDF algorithm achieved highest accuracy for set of sample that contain 10, 50 and 100 features with accuracy value 99.9%, 99.8% and 99.7 respectively. TF-IDF algorithm obtained 99.9%, 99.7% and 95.5% accuracy value for the same set of sample.

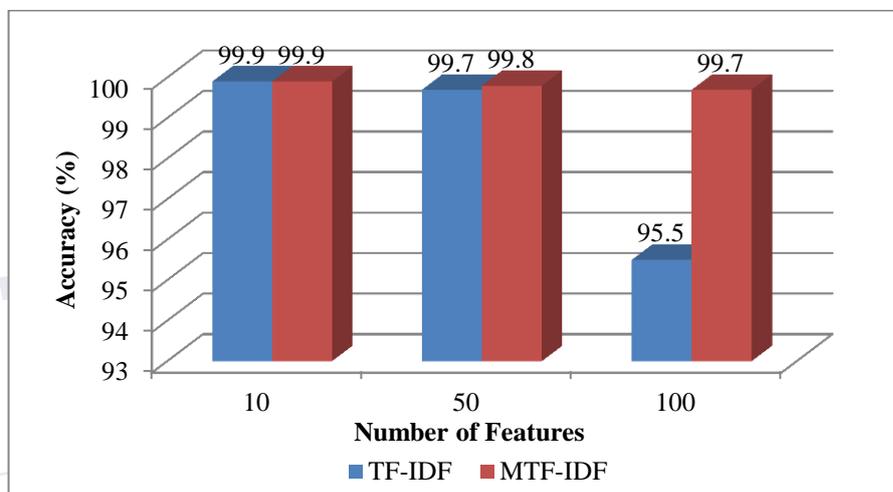


Figure 4.7: Accuracy value using different number of API call based feature tested on Random Forest

Meanwhile, Figure 4.8 shows the accuracy value based on different number of dangerous permission based features. MTF-IDF algorithm achieved highest accuracy for set of sample that contain 5, 10 and 15 features with accuracy of 99.9%, 99.8% and 99.6% respectively. TF-IDF algorithm obtained 99.8%, 99.4% and 98.6% accuracy for the same set of sample. From those results, the number of features used could affect the accuracy value in malware detection. The smallest number of features used, the higher accuracy value for malware detection. However, MTF-IDF successfully achieved better accuracy value using any number of features used compare to TF-IDF.

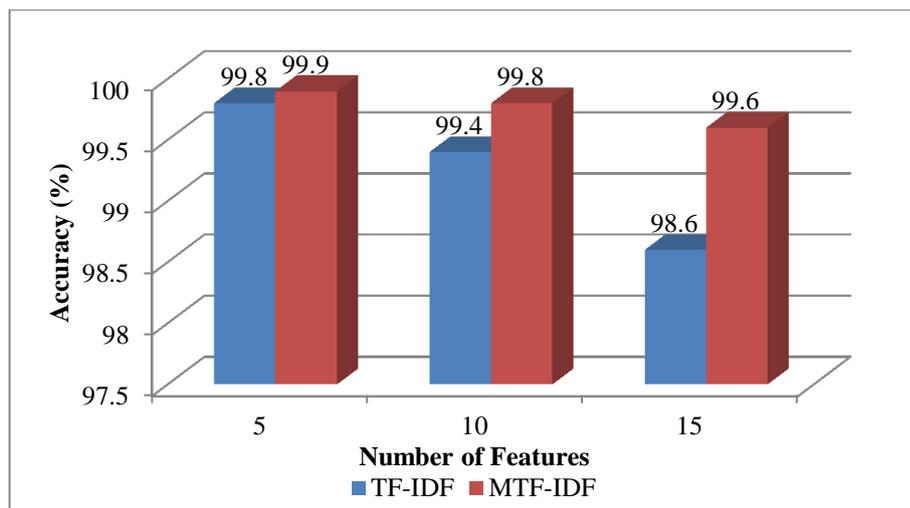


Figure 4.8: Accuracy value using different number of dangerous permission based feature tested on Random Forest

Table 4.4 and Table 4.5 show TP and FP rates for TF-IDF and MTF-IDF using different number of features for API call and dangerous permission respectively. Both TF-IDF and MTF-IDF algorithms showed lower FP rate value. Table 4.4 shows MTF-IDF algorithm achieved high TP rate which is 99.9% value when tested using 10 features of API call based feature. However, TF-IDF achieved highest TP value when tested using 50 API call based feature with 99.7%. This shows that MTF-IDF algorithms able to detect malware better using 10 API call based feature compared to TF-IDF algorithm.

Table 4.4: True positive (TP) and false positive (FP) rates on API call based feature tested on Random Forest algorithm

Feature	Experiment	TP Rate	FP Rate
10	TF-IDF	0.900	0.009
	MTF-IDF	0.999	0.001
50	TF-IDF	0.997	0.426
	MTF-IDF	0.999	0.001
100	TF-IDF	0.955	0.072
	MTF-IDF	0.968	0.035

While Table 4.5 shows MTF-IDF algorithm achieved high TP rate which is 99.9% value when tested using 5 features of dangerous permission based feature. TF-

IDF achieved 99.8 % TP value when tested using 5 dangerous permission based feature. This shows that MTF-IDF algorithm able to achieve higher TP rate using MTF-IDF algorithm compared to TF-IDF algorithm.

Table 4.5: True positive (TP) and false positive (FP) rates on dangerous permission based feature tested on Random Forest algorithm

Feature	Experiment	TP Rate	FP Rate
5	TF-IDF	0.998	0.007
	MTF-IDF	0.999	0.001
10	TF-IDF	0.994	0.003
	MTF-IDF	0.998	0.001
15	TF-IDF	0.986	0.08
	MTF-IDF	0.994	0.001

In this section, Android malware detection analysis was run based on the highest MTF-IDF and TF-IDF value. From this experiment, MTF-IDF is effective either using small or large number of features during malware detection process. The results show that the MTF-IDF can detect malware accurately and has higher true positive rates as compared to TF-IDF algorithm.

4.3.3 Integration Different Types of Feature

Table 4.6 shows 30 features used in integration of different types of feature, which are 15 API call based feature and 15 dangerous permission based feature. From each algorithm, the highest 15 features has been selected for API call and 15 for dangerous permission. In order to maintain the equality among two types of features, the number of features used for API call and dangerous permission are same.

Table 4.6: List of features used

	Features	Explanation
Application Program Interface (API) call based feature	AccountManager_addOnAccountsUpdatedListener	Notified listener whenever user or abstractaccountauthenticator (class) made changes to accounts.
	Activity_startActivity	Launch a new activity.
	ContentResolver_query	Query the given Uniform Resource Identifier (URI), returning a cursor (interface) over the result set with support for cancellation.
	Context_sendBroadcast	Broadcast the given intent to all interested broadcastreceivers (class), allowing an optional required permission to be enforced.
	Context_startActivity	Launch a new activity.
	Context_startService	Request that a given application service be started.
	PackageManager_setComponentEnabledSetting	Set the enabled setting for a package component (activity, receiver, service, provider).
	LocationManager_getBestProvider	Returns the name of the provider that best meets the given criteria.
	PowerManager_WakeLock_acquire	Acquires the wake lock.
	SmsManager_sendTextMessage	Send a text based SMS using.
	WebView	A View that displays web pages.
	URLConnection	A urlconnection with support for HTTP-specific features.
	Net_Socket	An endpoint for communication between two machines.
	URL_openConnection	Returns a urlconnection (class) instance that represents a connection to the remote object referred to by the URL.
URL_openStream	Opens a connection to this URL and returns an inputstream for reading from that connection.	

Table 4.6 (continue)

	Features	Explanation
Dangerous permission based feature	ACCESS_COARSE_LOCATION	Allows an application to access approximate location.
	ACCESS_FINE_LOCATION	Allows an application to access precise location.
	CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
	CAMERA	Required to be able to access the camera device.
	GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service.
	PROCESS_OUTGOING_CALLS	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
	READ_CONTACTS	Allows an application to read the user's contacts data.
	READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
	READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any PhoneAccounts registered on the device.
	RECEIVE_SMS	Allows an application to receive SMS messages.
	RECORD_AUDIO	Allows an application to record audio.
	SEND_SMS	Allows an application to send SMS messages.
	WRITE_CALENDAR	Allows an application to write the user's calendar data.
	WRITE_CONTACTS	Allows an application to write the user's contacts data.
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.	

Figure 4.9 shows number of malware detected based on integration of Android features. Number of malware detected for 200 samples is 66 using TF-IDF and 76 using MTF-IDF. Number of malware detected for both set of sample

increased with the increase in number of samples. The increase in number of malware detected between 200 samples and 1000 samples is 340 for TF-IDF and 376 for MTF-IDF algorithm.

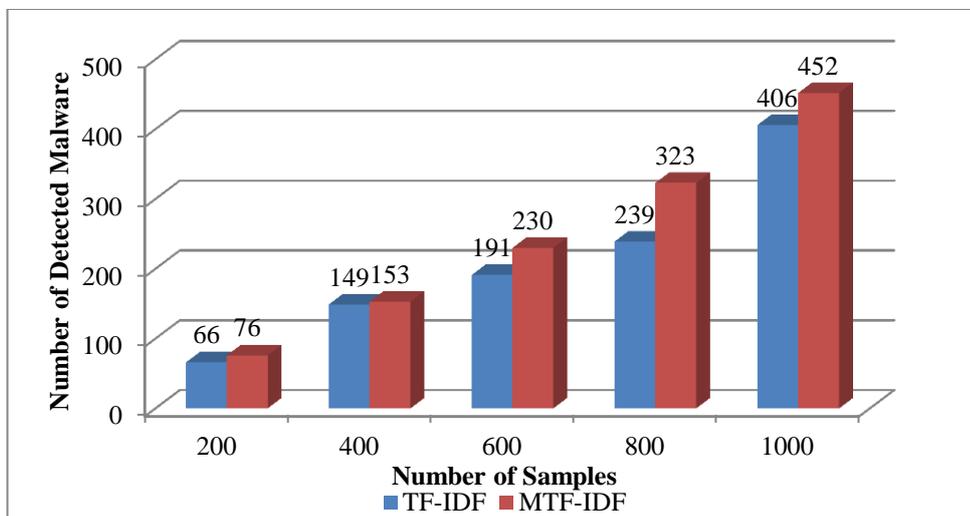


Figure 4.9: Number of malware detected on integrated features using Random Forest algorithm

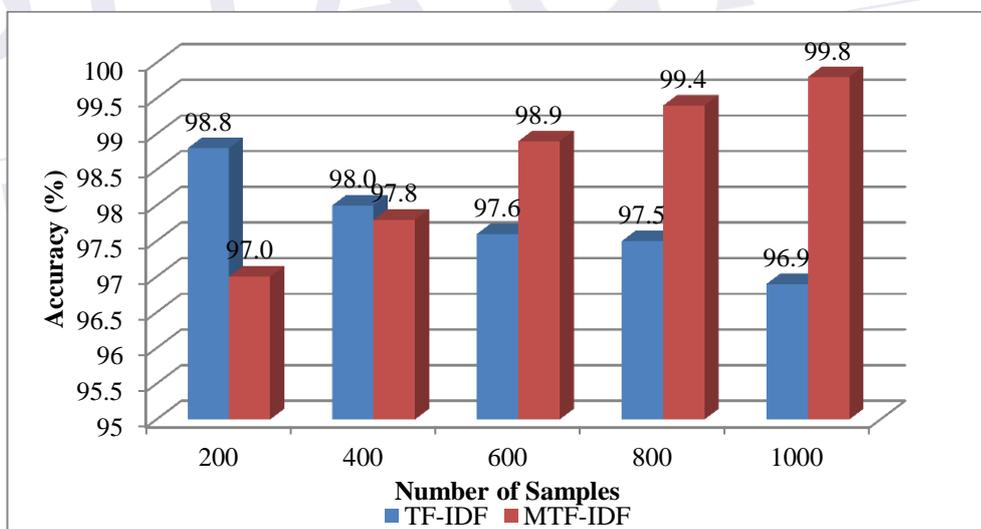


Figure 4.10: Performance comparison of accuracy using combination of features using Random Forest

Figure 4.10 shows the accuracy value based on integrated features. TF-IDF algorithm achieved highest accuracy for 200 and 400 samples of data with 98.8% and 98.0% accuracy respectively. Meanwhile, MTF-IDF algorithm obtained highest

accuracy for contain 600, 800 and 1000 samples set of sample with accuracy of 98.9%, 99.4% and 99.8% respectively. The result shows that feature selected by MTF-IDF algorithm is well diverse for different sample size either small or big sample of data. Feature selected can achieve goal accuracy value for all sample size for integrated features when tested on Random Forest algorithm.

Table 4.7: Performance comparison of accuracy based on algorithm using combination of API call and dangerous permission

Machine Learning Algorithm	Experiment	Algorithm	
		TF-IDF	MTF-IDF
Random Forest	200	98.0%	97.0%
	400	97.5%	97.8%
	600	98.8%	99.8%
	800	96.9%	99.4%
	1000	97.6%	98.9%
Decision Table	200	93.5%	93.5%
	400	95.5%	93.6%
	600	95.1%	98.2%
	800	94.0%	98.1%
	1000	95.0%	99.0%
Adaboost	200	98.0%	97.0%
	400	98.0%	98.3%
	600	97.8%	98.3%
	800	95.0%	98.0%
	1000	94.2%	98.4%

Table 4.7 shows the accuracy value based on different algorithms. 1000 samples of MTF-IDF algorithm achieved highest accuracy for all three algorithms which are Random Forest, Decision Table and Adaboost with accuracy of 98.9%, 99.0% and 98.4% respectively. Meanwhile, accuracy recorded for TF-IDF are 98%, 95% and 97.6% respectively.

Table 4.8 shows TP rate and FP rate for TF-IDF and MTF-IDF algorithms based on integrated features. For each set of sample, the TP rate is higher than FP rate in both experiments. TF-IDF algorithm performed better for small sample size that is 200 samples with 98% compared to MTF-IDF algorithm with 97%. However, MTF-IDF algorithm gained higher TP rate for 400, 600, 800 and 1000 samples, and

has higher TP rate compared to TF-IDF algorithm. The result shows that the proposed feature selection algorithm (MTF-IDF) is capable in detecting malware using various kinds of feature and different data size on machine learning algorithm.

Table 4.8: True positive (TP) and false positive (FP) rate based on integration of feature using Random Forest

Sample	Experiment	TP Rate	FP Rate
200	TF-IDF	0.980	0.041
	MTF-IDF	0.970	0.044
400	TF-IDF	0.975	0.038
	MTF-IDF	0.978	0.038
600	TF-IDF	0.988	0.015
	MTF-IDF	0.998	0.004
800	TF-IDF	0.969	0.037
	MTF-IDF	0.994	0.007
1000	TF-IDF	0.976	0.026
	MTF-IDF	0.989	0.012

4.4 Chapter Summary

Modified Term Frequency-Inverse Document Frequency (MTF-IDF) is proposed in feature selection during Android malware detection. Using feature selection, only helpful features selected during Android malware analysis. Noise features has been removed based on the weighted value. The proposed framework is tested against existing algorithm (TF-IDF) using different types of experimental setup that are different number of sample, different number of features and integration of features. Results made between the modified and existing feature selection algorithms were compared and only features with higher value were used in the analysis. The analysis results showed that feature selection using MTF-IDF algorithm can improve malware detection analysis. MTF-IDF algorithm proved that it is effective in detecting Android malwares either using various kinds of feature or various kinds of sample size. The next chapter discuss about the conclusion of this research and future works that can be done for improvements.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Introduction

This chapter concludes the whole research based on the performance of the results obtained. The previous chapter presents the results of the proposed algorithm feature selection using Modified Term Frequency-Inverse Document Frequency (MTF-IDF). The proposed method aims to determine the feature matrix for detecting Android malware using two types of Android features, which are dangerous permission (static feature) and Application Program Interface (API) call (dynamic feature).

The proposed framework is evaluated using DREBIN (Arp *et al.*, 2014) sample. Each sample has ran sample functions to verify its features, perform weighted based feature selection (MTF-IDF or TF-IDF), perform sample clustering using K-means to class the sample to either malware or benign and lastly construct feature matrix. The proposed weighted based feature selection technique is modified based on the TF-IDF algorithm where this approach focused on both sample and feature.

Modified Term Frequency-Inverse Document Frequency (MTF-IDF) algorithm give more focus on both sample and features to give correct weight value to some features. The proposed algorithm considered features based on its level of importance where weight given based on number of features involved in the sample. The MTF-IDF algorithm consists of three steps. First, each incoming sample ran functions to identify normalized Term Frequency (TF) for each feature involved. Next, Inverse Document Frequency (IDF) of Android sample for MTF-IDF was identified. The related best features in the sample are selected using weight and priority ranking process using K-means. This ensures that only important malware

features are selected in the Android application sample. Compared to TF-IDF, IDF only focuses on feature and not sample. Android malware detection using MTF-IDF algorithm achieved high accuracy compared to TF-IDF. Finally weight calculation using MTF-IDF for each sample was performed to determine feature matrix.

In evaluation of the proposed algorithm, Random Forest (Breiman, 2001) was used as main classification algorithm for machine learning algorithm, besides Decision Table and Adaboost. These three algorithms were selected because of their similar capabilities in detecting missing class value, nominal class, numeric class and binary class. Waikato Environment for Knowledge Analysis (WEKA) was used to measure the effectiveness of the feature selection approach (Hall *et al.*, 2009).

In order to measure the effectiveness of the feature selection approach, the performance of the proposed MTF-IDF algorithm were evaluated based on accuracy value. The proposed algorithm was tested using different types of experimental setup namely different number of sample, different number of features and combination of feature. The results showed that feature selection using MTF-IDF can improve malware detection analysis. This research demonstrated that the use of different types of feature or different size of sample not affect MTF-IDF effectiveness in Android malware detection. This chapter summarizes the outcome of this research and recommendations for possible research directions.

5.2 Contribution of the Research

Research on feature selection using weighted based technique is the easiest way to detect Android malware. This research is effective to analyze large number of features and reduce irrelevant features that may cause false results in detecting Android malware. Android malware detection can be run efficiently using feature selection via machine learning algorithm. Selecting the most useful subset of features from huge number of Android features can remove noisy and irrelevant data from sample collected and lead to more accurate results.

First contribution of this research is to propose feature selection technique using Term Frequency-Inverse Document Frequency (TF-IDF) for Android malware detection. This research improves Android malware detection by using the TF-IDF to select related best features in the sample using weight and priority ranking process.

This increases the effect of important malware features selected in the Android application sample.

Second contribution of this research is to develop a modified TF-IDF (MTF-IDF) algorithm to select the best features that could improve accuracy of Android malware detection. TF-IDF algorithm is widely used in text classification research and commonly used in text classification because it is richer and more successful representation for retrieval and categorization purposes. Yet, TF-IDF is unaware to the class labels in the sample, which can lead to inappropriate scaling for some features. This research modified TF-IDF in order to achieve ideal scaling, especially on scaling of both features and samples at the same time.

Third contribution of this research is to evaluate the performance of the proposed MTF-IDF algorithm based on accuracy. A few set of sample were produced to research the sensitivity of the algorithm. The algorithm was tested based on three requirements; different number of sample size, different number of features used and integration of different types of feature. Using different types of experimental setup, MTF-IDF proved that it is effective for Android malware detection for various kinds of feature or various kinds of sample size. As a conclusion, feature selection using MTF-IDF algorithm proved that it is effective in detecting Android malwares either based on a series of experiment.

5.3 Recommendations for Future Work

There are chances to improve the proposed algorithm and discover other potential issues. Thus, several recommendations are given below:

1. This research only focused on Application Program Interface (API) call and dangerous permission based feature. The proposed algorithm only studied Android malware characteristics using these features. So, it is recommended to perform further research not only for API call and dangerous permission, but the whole features applied on Android. All potential features in Android such as activity or Uniform Resource Locator (URL) should be explored to improve Android malware detection.
2. Comparison with other feature selection techniques. This research only focused on weighted based techniques (TF-IDF and MTF-IDF). Comparing and

researching other feature selection techniques could give high impact to the analysis and help to improve Android malware detection.

5.4 Concluding Remarks

This research proposed feature selection using Modified Term Frequency-Inverse Document Frequency (MTF-IDF) algorithm to enhance Android malware detection. The proposed algorithm improves Android malware detection by selecting best features in the samples using weight and achieved ideal scaling which focuses on of both features and samples at the same time. The importance of feature selection algorithm, such as MTF-IDF and TF-IDF, has been experimentally justified using machine learning algorithm. The proposed feature selection approach and existing approaches are compared using performance accuracy. MTF-IDF algorithm proved that it is an effective Android malware detection algorithm to cater for various kinds of feature or different kinds of sample size.



PTTA UTHM
PERPUSTAKAAN TUNKU TUN AMINAH

REFERENCES

- A Hamid, I. R. (2015). *Phishing Detection and Traceback Mechanism* (techreport). Deakin University.
- Abawajy, J., & Kelarev, A. (2017). Iterative Classifier Fusion System for the Detection of Android Malware. *IEEE Transactions on Big Data*, *PP(99)*, 1. article. <http://doi.org/10.1109/TBDATA.2017.2676100>
- Adib Azhar, M., Mohd Saudi, M., Ahmad, A., & Abu Bakar, A. (2019). Detection of Social Media Exploitation via SMS and Camera. *International Journal of Interactive Mobile Technologies (iJIM)*, *13(4)*, 61. <http://doi.org/10.3991/ijim.v13i04.10521>
- Afonso, V. M., de Amorim, M. F., Grégio, A. R. A., Junquera, G. B., & de Geus, P. L. (2015). Identifying Android malware using dynamically obtained features. *Journal of Computer Virology and Hacking Techniques*, *11(1)*, 9–17. article.
- Alejandro, M., Lara-Cabrera, R., & Camacho, D. (2019). Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset. *Information Fusion*, *52*, 128–142. article.
- Android Developers. (2018a). Application Fundamentals. <https://developer.android.com/index.html>. misc, Android Developer. Retrieved from <https://developer.android.com/guide/components/fundamentals>
- Android Developers. (2018b). Permissions [misc]. Retrieved December 22, 2018, from <https://developer.android.com/guide/topics/permissions/index.html>
- Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., & Rieck, K. (2014). DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *NDSS*. inproceedings.
- Aung, Z., & Zaw, W. (2013). Permission-Based Android Malware Detection. *International Journal of Scientific & Technology Research*.

- Bhattacharya, A., & Goswami, R. T. (2017). Comparative Analysis of Different Feature Ranking Techniques in Data Mining-Based Android Malware Detection. In S. C. Satapathy, V. Bhateja, S. K. Udgata, & P. K. Pattnaik (Eds.), *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications: FICTA 2016, Volume 1* (pp. 39–49). inbook, Singapore: Springer Singapore. http://doi.org/10.1007/978-981-10-3153-3_5
- Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowddroid: Behavior-based Malware Detection System for Android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (pp. 15–26). inproceedings, New York, NY, USA: ACM. <http://doi.org/10.1145/2046614.2046619>
- Castillo, C. A., & others. (2011). Android malware past, present, and future. *White Paper of McAfee Mobile Security Working Group, 1*, 16. article.
- Chen, Z., & Liu, B. (2018). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3), 1–207. article.
- Darcey, L., & Conder, S. (2012). *Android Wireless Application Development Volume I: Android Essentials*. book, Addison-Wesley.
- Feizollah, A., Anuar, N. B., Salleh, R., Amalina, F., Ma'arof, R. R., & Shamshirband, S. (2013). A Study of Machine Learning Classifiers for Anomaly-Based Mobile Botnet Detection. *Malaysian Journal of Computer Science*.
- Feizollah, A., Anuar, N. B., Salleh, R., & Wahab, A. W. A. (2015). A Review on Feature Selection in Mobile Malware Detection. *Digital Investigation*. <http://doi.org/10.1016/j.diin.2015.02.001>
- Felt, A. P., Finifter, M., Chin, E., Hanna, S., & Wagner, D. (2011). A Survey of Mobile Malware in the Wild. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (pp. 3–14). inproceedings, New York, NY, USA: ACM. <http://doi.org/10.1145/2046614.2046618>
- Feng, Y., Anand, S., Dillig, I., & Aiken, A. (2014). Semantics-based detection of android malware through static analysis. *SIGSOFT FSE, Apposcopy*. article.
- Forman, G. (2008). BNS Feature Scaling: An Improved Representation Over TF-IDF for SVM Text Classification. *Proceedings of the 17th ACM Conference on*

<http://doi.org/10.1145/1458082.1458119>

- Freund Y., & S. R. E. (1995). A Decision-Theoretic Generalization Of On-Line Learning and An Application To Boosting. *In Computational Learning Theory*, 33–37. article.
- G Data. (2016). *G Data Releases Mobile Malware Report For The Fourth Quarter Of 2015* (webpage). Retrieved from <https://www.gdata-software.com/g-data/newsroom/news/article/g-data-releases-mobile-malware-report-for-the-fourth-quarter-of-2015> (Search Date 19/4/2016)
- Gandhewar, N., & Sheikh, R. (2011). Google Android: An Emerging Software Platform For Mobile Devices. *International Journal on Computer Science & Engineering, Supplement*(12), 12–18. <http://doi.org/10.1007/s003810050241>
- Gianazza, A., Maggi, F., Fattori, A., Cavallaro, L., & Zanero, S. (2014). PuppetDroid: {A} User-Centric {UI} Exerciser for Automatic Dynamic Analysis of Similar Android Applications. *CoRR, abs/1402.4*. article. Retrieved from <http://arxiv.org/abs/1402.4826>
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1), 10–18. article.
- Hamid, I. R. A., & Abawajy, J. H. (2014). An Approach for Profiling Phishing Activities. *Computers and Security*. <http://doi.org/10.1016/j.cose.2014.04.002>
- Hamid, I. R. A., Khalid, N. S., Abdullah, N. A., Rahman, N. H. A., & Wen, C. C. (2017). Android Malware Classification Using K-Means Clustering Algorithm. *IOP Conference Series: Materials Science and Engineering*, 226, 12105. <http://doi.org/10.1088/1757-899X/226/1/012105>
- Jang, J., Yun, J., Mohaisen, A., Woo, J., & Kim, H. K. (2016). Detecting and Classifying Method Based on Similarity Matching of Android Malware Behavior with Profile. *SpringerPlus*, 5(1), 1. article.
- Kakavand, M., Dabbagh, M., & Dehghantanha, A. (2018). Application of Machine Learning Algorithms for Android Malware Detection. article.
- Karbab, E. B., Debbabi, M., Alrabaee, S., & Mouheb, D. (2016). DySign: dynamic fingerprinting for the automatic detection of Android malware. In *Malicious and Unwanted Software (MALWARE), 2016 11th International Conference on* (pp. 1–8). inproceedings.

- Kaspersky Lab. (2019). Mobile Malware Evolution 2018. webpage. Retrieved from <https://securelist.com/mobile-malware-evolution-2018/89689/>
- Lee, H.-M., Wu, D.-J., Mao, C.-H., & Wei, T.-E. (2014). Method and System for Detecting Malicious Application. misc, Google Patents.
- Li, W., Ge, J., & Dai, G. (2016). Detecting Malware for Android Platform: An SVM-Based Approach. In *Proceedings - 2nd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2015 - IEEE International Symposium of Smart Cloud, IEEE SSC 2015*. <http://doi.org/10.1109/CSCloud.2015.50>
- Mas'ud, M. Z., Sahib, S., Abdollah, M. F., Selamat, S. R., & Yusof, R. (2014). Analysis of Features Selection and Machine Learning Classifier in Android Malware Detection. *2014 International Conference on Information Science & Applications (ICISA)*, 1–5. <http://doi.org/10.1109/ICISA.2014.6847364>
- Mazlan, N. H., & Hamid, I. R. A. (2018a). Evaluation of Feature Selection Algorithm for Android Malware Detection. *International Journal of Engineering & Technology*, 7(4.31), 311–315. article.
- Mazlan, N. H., & Hamid, I. R. A. (2018b). Using Weighted Based Feature Selection Technique for Android Malware Detection. In K. J. Kim & N. Joukov (Eds.), *Mobile and Wireless Technologies 2017* (pp. 54–64). Singapore: Springer Singapore.
- McWilliams, G., Sezer, S., & Yerima, S. Y. (2014). Analysis of Bayesian Classification-Based Approaches for Android Malware Detection. *IET Information Security*. <http://doi.org/10.1049/iet-ifs.2013.0095>
- Meng, H., Thing, V. L. L., Cheng, Y., Dai, Z., & Zhang, L. (2018). A Survey of Android Exploits in The Wild. *Computers and Security*, 76, 71–91. <http://doi.org/10.1016/j.cose.2018.02.019>
- Milosevic, N., Dehghantanha, A., & Choo, K.-K. R. (2017). Machine learning aided Android malware classification. *Computers & Electrical Engineering*, 61, 266–274. article.
- Raheel Shaikh. (2018). Feature Selection. <https://towardsdatascience.com/>. misc, Towards Data Science. Retrieved from <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
- Rao, V., & Hande, K. (2017). A comparative study of static, dynamic and hybrid analysis techniques for android malware detection. *Int. J. Eng. Dev.*

- Res.(IJEDR)*, 5, 1433–1436. article.
- Samat, N. A. (2017). *An Improved Imputation Method Based on Fuzzy C-Means and Particle Swarm Optimization for Treating Missing Data* (phdthesis). Universiti Tun Hussein Onn Malaysia.
- Sami, A., Yadegari, B., Rahimi, H., Peiravian, N., Hashemi, S., & Hamze, A. (2010). Malware Detection Based on Mining API Calls. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 1020–1025). New York, NY, USA: ACM. <http://doi.org/10.1145/1774088.1774303>
- Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). “Andromaly”: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1), 161–190. article.
- Shabtai, A., Moskovitch, R., Elovici, Y., & Glezer, C. (2009). Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A state-of-the-art survey. *Information Security Technical Report*, 14(1), 16–29. article.
- Statista. (2019). Desktop vs Mobile Market Share Worldwide. misc, Statista. Retrieved from <http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/2018>
- Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A. (2014). Evolution, Detection and Analysis of Malware for Smart Devices. *IEEE Communications Surveys Tutorials*, 16(2), 961–987. article. <http://doi.org/10.1109/SURV.2013.101613.00077>
- Tian, R. (2011). An Integrated Malware Detection and Classification System.
- Trend Micro. (2012). A Brief History of Mobile Malware. webpage. Retrieved from <https://countermeasures.trendmicro.eu/wp-content/uploads/2012/02/History-of-Mobile-Malware.pdf> (Search Date 19/4/2016)
- Vidas, T., Votipka, D., & Christin, N. (2011). All Your Droid Are Belong to Us: A Survey of Current Android Attacks. In *WOOT* (pp. 81–90). inproceedings.
- We Are Social. (2019). Global Digital Report 2018, World’s Internet Users Pass The 4 Billion Mark. misc. Retrieved from <https://digitalreport.wearesocial.com>
- WIRED. (2019). An Android Vulnerability Went Unfixed for Over Five Years. misc, CNMN Collection. Retrieved from <https://www.wired.com/story/android-vulnerability-five-years-fragmentation/>
- Zhang, M., Duan, Y., Yin, H., & Zhao, Z. (2014). Semantics-Aware Android

Malware Classification Using Weighted Contextual API Dependency Graphs.
In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and
Communications Security* (pp. 1105–1116). inproceedings, New York, NY,
USA: ACM. <http://doi.org/10.1145/2660267.2660359>

