Journal of Wireless Networking and Communications 2015, 5(2A): 6-12
DOI: 10.5923/c.jwnc.201501.02

Recognizing API Features for Malware Detection Using Static Analysis

Saidah Mastura A. Ghani*, Mohd Faizal Abdollah, Robiah Yusof, Mohd Zaki Mas'ud

Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, UTeM, Malaysia

Abstract Android is one of the mobile operating system. Rapidly increasing of devices which used android as platform, make the Android is the best target to cybercriminal. Besides, Android offers many applications and these applications can get from other than Google Play. This make the cybercriminal easier develop malware and easily spread the malware to user. Malware can be made by injecting the malicious code into benign applications. Therefore, this research will use static analysis technique. Static analysis technique will extracted the benign and malware application to get their source code. The source code in benign and malware will be compared and categorized into API and manager classes. Then the most frequent API and manager class used in malware will be detected.

Keywords Android, Static analysis, API, Manager classes, Malware

1. Introduction

Mobile devices such as smartphone and tablet computer gained popularity among users nowadays. Users use their mobile device for many of the same purpose as desktop which to browse the Internet, make the online banking, update status in social networks, search location and others.

Android is one of the mobile operating system. Android is an open source mobile operating system developed by Open Handset Alliance (OHA) which led by Google. Android is based on a modified Linux 2.6 kernel [1]. Android is designed primarily for touch screen mobile devices.

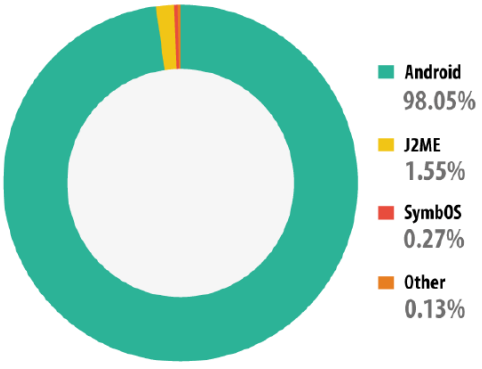

Figure 1. Distribution of Mobile Malware in 2013 by Platform [11]

In figure 1, it shows the distribution graph of mobile malware detected in 2013 by platform. The graph shows that the most infected platform is Android.

There are three factors needed before an increase of mobile malware will occur, which are, an open platform, a ubiquitous platform and the cybercriminal motivations [13].

Android is an open source mobile platform. Besides, Android allows users to develop Android application and the application can be published without review by third party [5]. Therefore, the cybercriminal take this opportunity to develop malware in Android platform.

Other than that, Android capability to run on different devices with different version exposes to varied security issues since the customization of Android security is done by device manufacturer [5]. Therefore, Android become target for cybercriminal because of Android is widely use by many products and make the Android is a ubiquitous platform.

Google Play, Android official market share, offers thousands of applications to users either the applications is free or paid. These applications including games, social networking, multimedia and etc. However, Android follows a *laissez-faire* philosophy, which means users can get Android applications from variety of source beyond the Google Play [2]. Because of this, cybercriminal can develop malware and distribute it to users easily.

Cybercriminal can easily develop a malware by injecting the malicious code in benign Android application. For this reason, this research will be conducted using static analysis technique.

Static analysis technique is using the reverse engineering method where the source code of application is extracted. This technique involves the automatic application code lookup which means, the required content is detect without running or testing the applications [3]. This makes the

* Corresponding author:
saidah0812@gmail.com (Saidah Mastura A. Ghani)
Published online at http://journal.sapub.org/jwnc

technique faster in detection of malware. Other than that, static analysis technique can get high detection rate and consume fewer resources [3].

This research will focus on Android Package Index (API) and manager classes. Android contains of 238 of APIs [4]. Under all these 238 APIs, there are many classes which rich of framework to help developers develop their application. But this research will focus on manager classes of APIs. The managers are provided in application framework layer [10]. Figure 2 shows the Android application framework layer and its blocks which contain managers.

Application framework layer is the most vulnerable layer in Android architecture as stated in figure 3. The vulnerability of application framework layer makes the layer is easier for cybercriminal to inject malicious code on it.

Besides application framework layer of Android contains most of users' sensitive information. Therefore, cybercriminal will used malware on this layer in order to get the user's information.

As example, Android.Tapsnake which pretending to be just a game of snake while at the background, the application is uploading the GPS (Global Positioning System) coordinates of the device every 15 minutes [13]. The coordinate is send back to criminal to locate the user's location.

Android device is allowed to determine current location via GPS, cell tower or wifi network [18]. The android.location is API used to determine current location [18]. In the Android.Tapsnake, the android.location API is used to activate the GPS.

Android.FakePlayer is another example of malware. This malware sends multiple messages to two short-codes, premium rate number [13]. This malware sends two messages at premium rate to two numbers which, the first number rate is approximately $3.50 and the second number rate is at $6 and makes it $13 each time the application is executed [13]. Once the messages are sent, user is billed but not realize about the sent messages since the messages are sent in the background without prompting the user [13].

In this scenario, the android.telephony API is called in the source code of the malware. In the android.telephony, there are SmsManager class which manage SMS (Short Message Service) operation such as sending data, text and pdu SMS messages [17]. In this malware, the SmsManager is used to send the messages to the two short-codes, premium rate number.

This paper is aim to perform static analysis in both malware and benign Android application. Then, from the extracted file, the source code of malware and benign will be compared and categorized. There are two categories, which are API and manager classes. The most API and manager class used in malware and benign will determine at the end of this research.

This paper is organized as follows: in section 2, the related works is discussed; in section 3, the methodology of this research is presented; in section 4 the result for this research is discussed; and finally in section 5, the conclusion of this research and the future work is presented.

2. Related Works

Android applications malware detection can be analyze by using two techniques which are static analysis technique and dynamic analysis technique. There are many researches about malware detection in Android applications using either dynamic analysis or static analysis technique or both.

In [6], [7] and [8], the dynamic analysis technique is used. Meanwhile in [3], [14] and [15] the static analysis technique is used.

In [6], permissions in manifest file are used. These permissions are then categorized into two groups, standard built-in permissions and non-standard built-in permissions. The Control Flow Graph (CFG) is then used to detect the malware.

In [7], some Android applications which suspected to be in Command and Control (C&C) Server are downloaded. Then a relationship graph using Gephi is constructed. Sensitive API invoking and declares Intent-filter are searched within the graph. A* algorithm is used to find a least-cost path in benign and malware from graph.

In [8], the flow of system calls of applications is monitored. The flow of applications system calls is monitored across the layer of Android architecture. The malicious flow is monitored to compare with the normal flow of Android applications.

In [3], Intents in Android applications are used to categorize the application either benign or malware. These Intents are got by extracting the Android application .apk file using APKTool to get the byte code. The custom C++ is then used to tag the feature. After that, another custom C++ is used to categorize. The tagged feature and categorization used to point out the features of malicious applications.

Figure 2. Android Application Framework Layer [1]

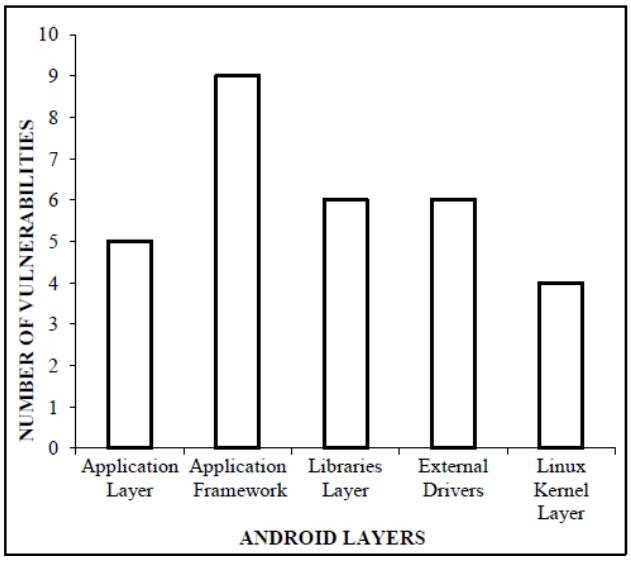

Figure 3. Vulnerabilities of Android Architecture Layer [5]

[14] is a research where to compare the 10-Fold Cross Validation Scheme, which often use in malware detection, with reality. The dataset used in this research were applications from Android Market, collecting known malware from Genome project and malware labelling from VirusTotal. The applications from dataset are extracted using static analysis. After that, the model is classified into known malware and testing set. Then, the classification validation scenario is done and grouped into 2 (two) scenarios which are: 10-Fold Cross validation and Validation in Wild.

In [15], machine learning is used to classify the malware applications. The .apk file of Android application is extracted to get the real permission required by application, and adopted the features for malware detection. A few sample features are used as train dataset by using K-Means clustering algorithm. After the train data is developed, the decision tree is used to each cluster to classify the malware applications.

Both dynamic and static analysis techniques are used in [16]. The static analysis is used by extracting the manifest file to get the permission source code used on applications. The manifest file was extracted and decrypted by using Android Asset Packaging Tool (aapt). Dynamic analysis is used in [16] to monitor the topology of input space of application. Self-Organization Map (SAM) is used in order to monitor the application input space. The input space then calculated using the Euclidean distance criterion. The results in [16] are based on Permission Protection Level: Normal, Dangerous, Signature and SignatureOrSystem.

DroidAPIMiner [12], used static analysis technique by analyze benign and malware using Androguard. DroidAPIMiner mining the API level features to detect the critical API calls, their package level information and their parameter. DroidAPIMiner used large of dataset of malware and benign. DroidAPIMiner identified the most frequently used APIs in malware.

The most frequently APIs used in benign and malware from dataset will identified. After that, the API calls that are exclusively invoked by third-party packages like advertisement is removed. If the API is frequently in both malware and benign, the data flow analysis is conducted. This data flow is used to recover the API parameter value and selected only the APIs that invoke dangerous values. Then, the most frequently APIs in malware is detected.

This research focused on APIs and manager classes. DroidAPIMiner went through every class and parameter of the APIs, but this research only focus on manager classes.

Rather than collected random malware and benign applications, this research compared the APIs used in the identical malware and benign applications. This comparison will gathered the most frequently used APIs in malware and benign applications. Then the APIs with manager classes will be further compared in both malware and benign applications.

3. Methodology

Figure 4 shows the methodology used in this research. This research compare the malware and benign applications which identically to each others. This means, both malware and benign applications in this research are identical but the malware applications were already injected by malicious code.

This methodology has two (2) phases which are feature extraction phase and feature comparison phase.

-Feature Extraction

In this phase, both malware and benign applications will be extracted by using Androguard, a reserve engineering tool. The antivirus will be injected to the benign applications before they are extracted. This is done to make sure the benign applications are really clean from the malware. The AVG antivirus is used since it is the most popular antivirus detector in [9].

-Feature Comparison

After the malware and benign applications are extracted, the source code between malware and benign applications will be compared. Then the differentiation of source code between malware and benign applications will be detected.

Then, the parameter is categorized into two categories. The categories are APIs and manager classes. There are 238 APIs in Android from level 1 API to level 22 API. The first categorization will detect all of the APIs involved in both malware and benign applications.

After that, the APIs classes which involved with manager classes will detect. The examples of API with manager classes are Telephony Manager, SmsManager, Power Manager, Connectivity Manager, and Notification Manager.

The results of malware and benign applications from both categories will be drawn using graphs. The frequency of APIs and manager classes categories will be compare from

benign and malware applications to detect the most frequently used API and manager classes by malware.

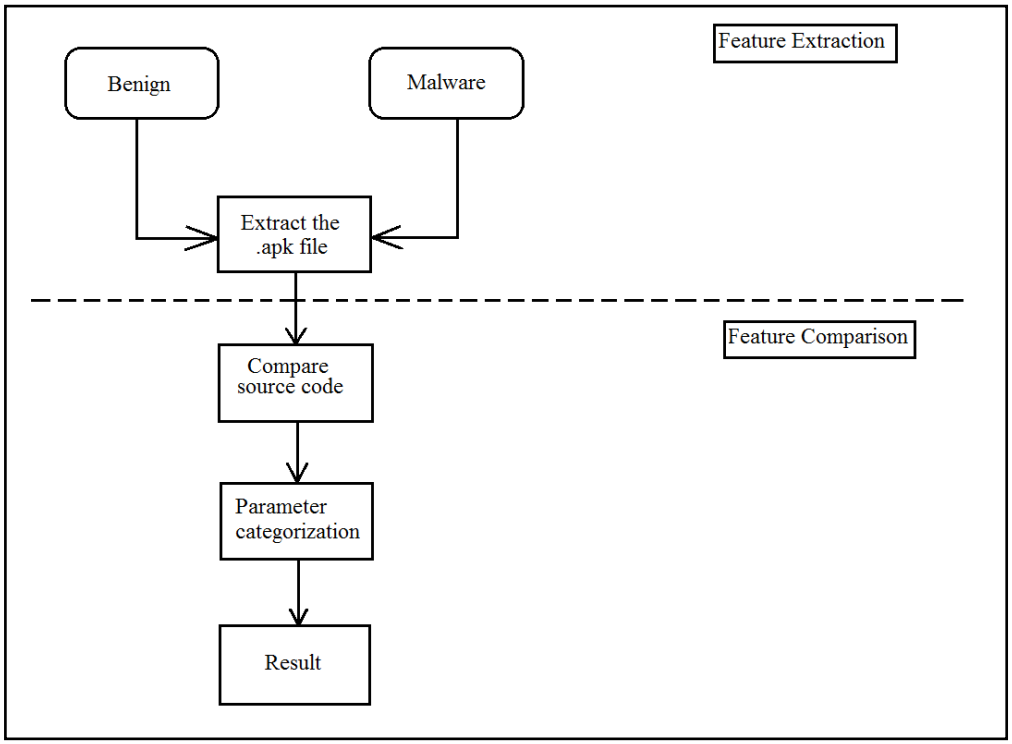

Figure 4. Research Methodology

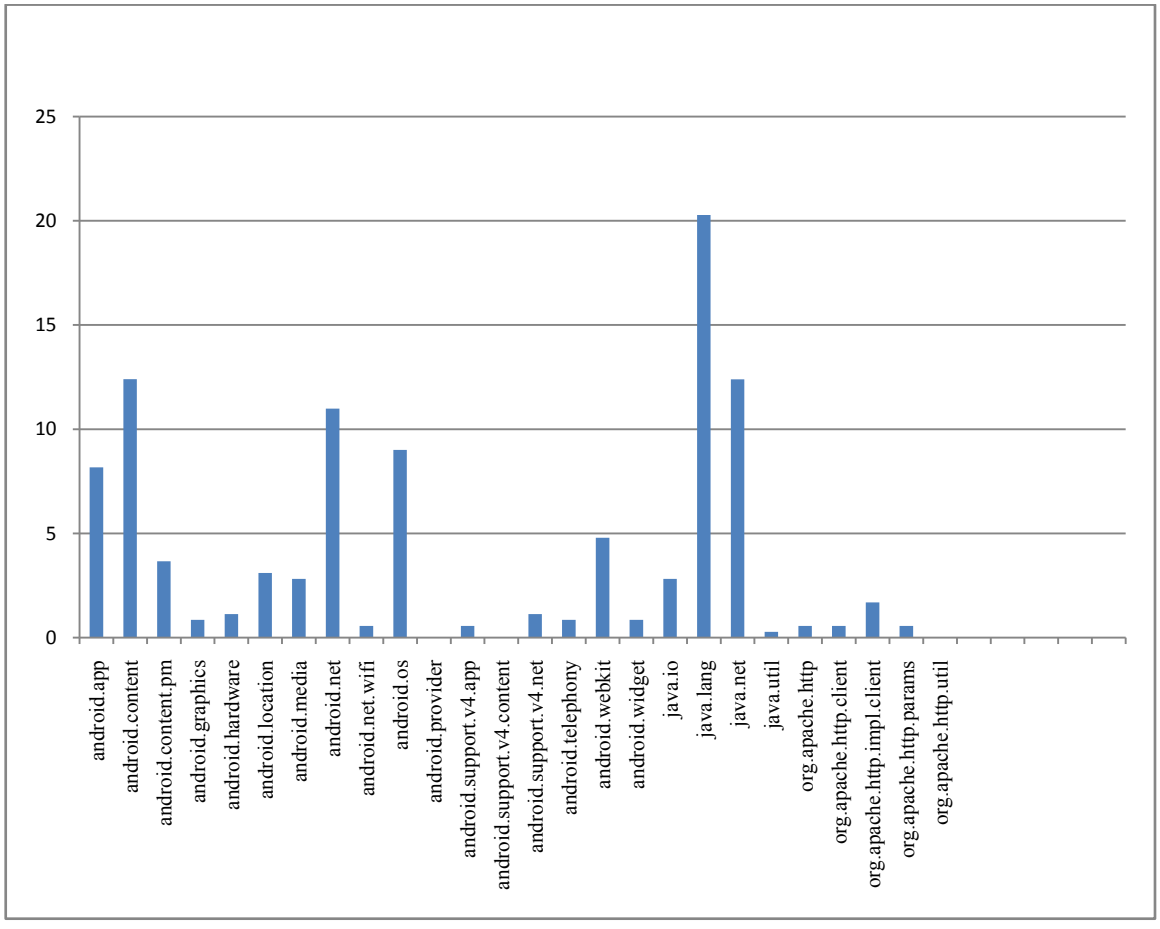

Figure 5. Result of APIs Used in Benign Applications

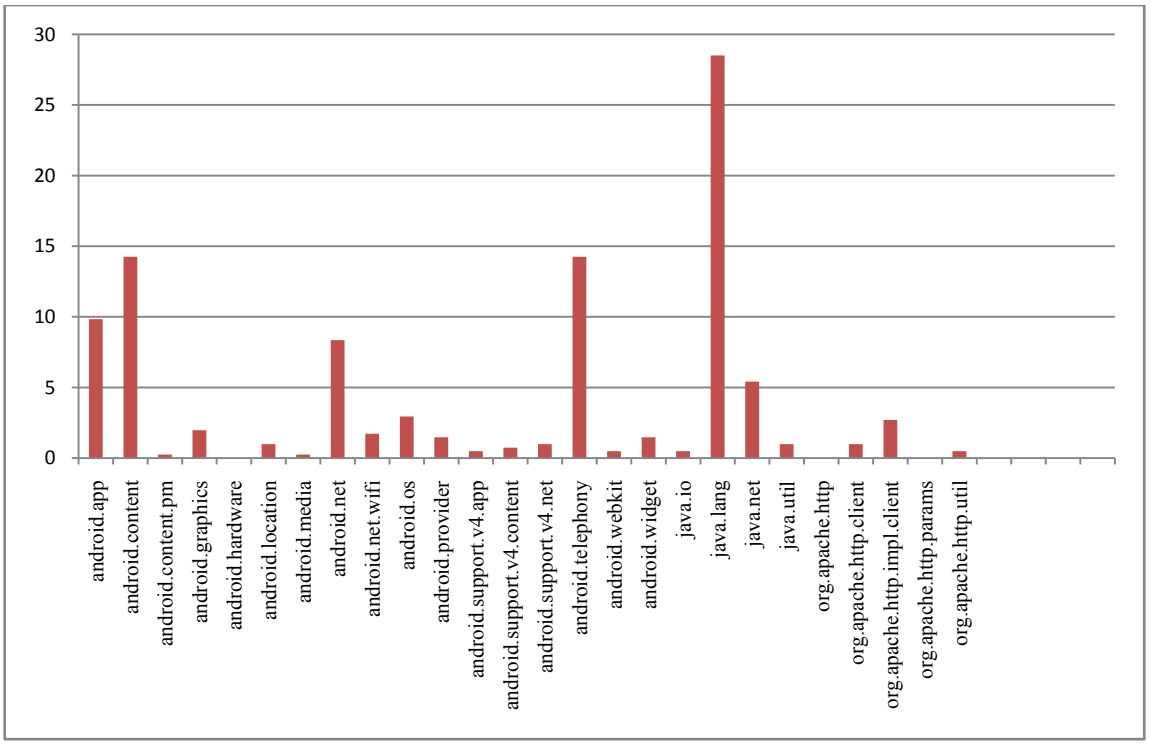

Figure 6. Result of APIs Used in Malware Applications

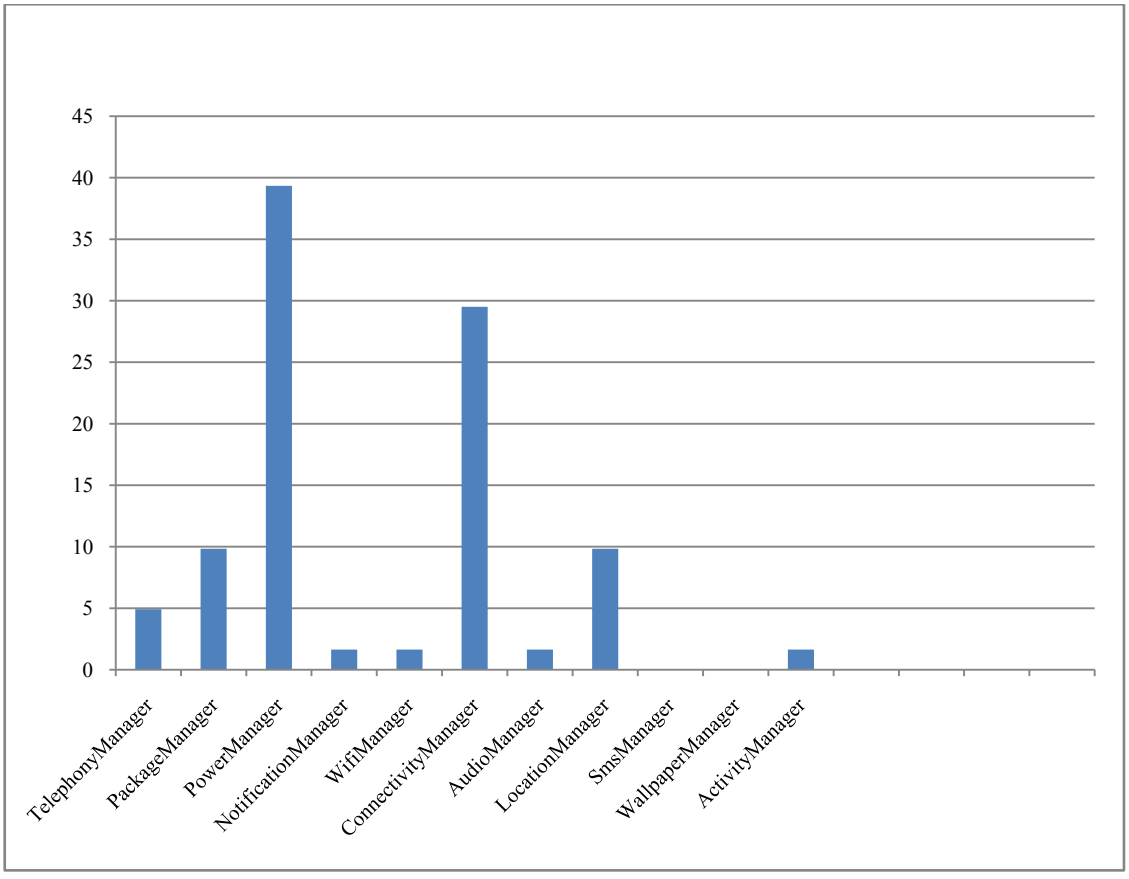

Figure 7. Result of Manager Classes Used in Benign Applications

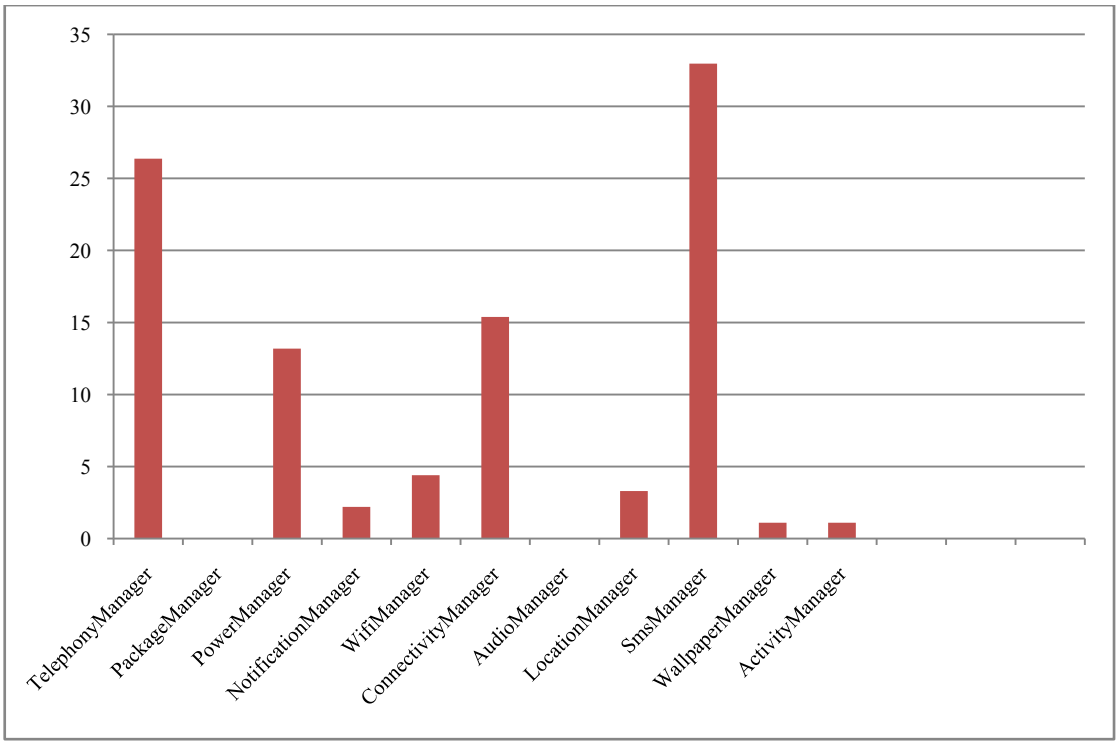

Figure 8. Result of Manager Classes Used in Malware Applications

4. Results

As the preliminary test, 7 benign and 7 malware applications are tested. The benign and malware used in this test are identical.

Table 1. Top 3 APIs Obtained From Result

API Category				
No.	Benign	(%)	Malware	(%)
1st	java.lang	20.28	java.lang	28.5
2nd	android.content	12.39	android.content	14.25
	java.net	12.39	android.telephony	14.25

Table 1 shows the top 3 APIs used by benign and malware applications from result. The java.lang API is the most frequently used API in both benign and malware applications. The android.content is also the second most API used for both benign and malware applications. However, the android.telephony is the second most used API in malware while in benign the second most API used is java.net (second place share the same percentage).

Table 2. Top 3 Manager Classes Obtained from Result

Manager Classes Category				
No.	Benign	(%)	Malware	(%)
1st	PowerManager	39.34	SmsManager	32.97
2nd	ConnectivityManager	29.51	TelephonyManager	26.37
3rd	LocationManager	9.84	ConnectivityManager	15.38
	PackageManager			

Table 2 shows the top 3 manager classes used by benign and malware applications from result. Power Manager is the most used manager class in benign applications while SmsManager is the most manager class used in malware applications.

From the result, the SmsManager and the Telephony Manager is the first and second top manager class used in malware. Both of these manager classes are the manager class of the android.telephony API. Although the android.telephony is the second most used API in malware, but its manager classes is the first and second manager class used by malware. Therefore, the android.telephony can be said as the API which often used by malware.

5. Conclusions and Future Work

From the results, it shown that, the SmsManager and Telephony Manager are the top two most manager classes used by malware. Both of these manager classes are the manager class in android.telephony API [17].

As conclusion, android.telephony with manager classes of SmsManager and Telephony Manager are the most used APIs and manager classes in malware applications.

Therefore, when used identical benign and malware applications, the difference of frequency of APIs and manager classes for both benign and malware can be detected. Besides, the most frequently used API and manager class too will be detected. The result in this research shows the relationship between the most used API and manager classes in malware.

As implication, the result from this research is very useful especially for Android applications developers to take some awareness when using the Android APIs and manager classes. Besides, the vulnerable of some APIs and manager classes can be determine since those APIs and manager classes are easily exploit by cybercriminal. Therefore, some security features can be enhanced in those APIs and manager classes.

As the future work, there will be more identical malware and benign applications will be used in this research. The machine learning tool will be used in this research to classify the APIs and manager classes of benign and malware.

ACKNOWLEDGEMENTS

Thank you to Universiti Teknikal Malaysia Melaka (UTeM) because gives me the opportunity to do this research and also provide the research facilities to me.

REFERENCES

[1] Dominique A. Heger, "Mobile Devices - An Introduction to the Android Operating Environment Design, Architecture, and Performance Implication", in DHTechnologies, Texas, USA, Tech.Report, 2012.

[2] Kindsight, "The Mobile Malware Problem", in A Kindsight White Paper, Ottawa, Canada, Tech.Report, 2012.

[3] Muhammad Zuhair Qadir, Atif Nisar Jilani and Hassan Ullah Sheikh, "Automatic Feature Extraction, Categorization and Detection of Malicious Code in Android Application", in Proceeding International Journal of Information and Network Security, vol.3, no.1, pp.12-17, 2014.

[4] Online Available: http://developer.android.com/reference/android/content/package-summary.html.

[5] Himanshu Shewale, Sameer Patil, Vaibhav Deshmukh and Pragya Singh, "Analysis of Android Vulnerabilities and Modern Exploitation Techniques", in ICTACT Journal on Communication Technology, vol.5, no.1, 2014.

[6] Justin Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection", in Intelligence and Security Informatics Conference, Odense, European, 2012.

[7] Luoshi Zhang, Yan Niu, Xiao Wu, Zhaoguo Wang and Yibo Xue, "A3: Automatic Analysis of Android Malware", in International Workshop on Cloud Computing and Information Security, 2013.

[8] Alessandro Armando, Alessio Merlo and Luca Verderama, "Security Issues in the Android cross-layer architecture", 2012.

[9] Heqing Huang, Kai Chen, Peng Liu, Sencun Zhu and Dinghao Wu, "Uncovering the Dilemmas on Antivirus Software Design in Modern Mobile Platforms", in Proceedings of the International Workshop on System Level Security Of Smartphones (SLSS 2014), Beijing, China, September 23, 2014.

[10] Stefan Brahler, "Analysis of the Android Architecture", Karlsruhe Institute of Technology, Tech. Rep, 2010.

[11] Victor Chebyshev and Roman Unuchek, "Mobile Malware Evolution: 2013", in Kapersky Report Mobile Malware Evolution, 2013.

[12] Yousra Aafer, Wenliang Du and Heng Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android", in Security and Privacy in Communication Networks, pp. 86-103, 2013.

[13] Eric Chin, "Motivations of Recent Android Malware", Symantec Security Response, Tech. Rep, 2011.

[14] Kevin Allix, Tegawende Bissyande, Quentin Jerome, Jacques Klein and Radu State, "Large-Scale Machine Learning-based Malware Detection: Confronting the "10-Fold Cross Validation" Scheme with Reality", in Conference on Data and Application Security and Privacy, San Antonio, Texas, USA, 2014.

[15] Zami Aung and Win Zaw, "Permission-Based Android Malware Detection", in International Journal of Scientific & Technology Research, vol.2, no.3, 2013.

[16] Chit La Pyae Myo Hein, "Permission Based Malware Protection Model for Android Application", in International Conference on Advances in Engineering and technology, Singapore, 2014.

[17] Online Available: http://developer.android.com/reference/android/telephony/package-summary.html.

[18] Online Available: http://www.vogella.com/tutorials/AndroidLocationAPI/article.html#locationapi_overview.