International Journal of Software Engineering and Technology (IJSET)
Vol. 1, No. 2, August 2016, pp. 59~68
ISSN: 2032-4038

Factors that Influence the Android Apps Permissions

Normi Sham Awang Abu Bakar, Iqram Mahmud
Department of Computer Science, International Islamic University Malaysia

Article Info

Article history:

Received Jun 12th, 2015
Revised Aug 20th, 2015
Accepted Aug 26th, 2015

Keyword:

Android Applications
Empirical Analysis
Sensitive Data
Permission Types
Statistics

ABSTRACT

The Android Market is the official (and primary) storefor Android applications. The Market provides users with average user ratings, user reviews, descriptions, screenshots, and permissions to help them select applications. Generally, prior to installation of the apps, users need to agree on the permissions requested by the apps; they are not given any other option. Essentially, users may not aware on some security issues that may arise from the permissions. Some apps request the right to manipulate sensitive data, such as GPS location, photos, calendar, contact, email and files. In this paper, we explain the sources of sensitive data, what the malicious apps can do to the data, and apply the empirical software engineering analysis to find the factors that could potentially influence the permissions in Android apps. In addition, we also highlight top ten most implemented permissions in Android apps and also analyse the permissions for the apps categories in Android.

Corresponding Author:

Normi Sham Awang Abu Bakar,
Department of Computer Science,
International Islamic University Malaysia,
Jalan Gombak, 53100 Kuala Lumpur, Malaysia.
Email: nsham@iium.edu.my

1. INTRODUCTION

The smart phone industry has grown very rapidly in the recent years. Android, developed by the Open Handset Alliance, is the most popular mobile platform for smart phones worldwide. The central repository for mobile applications, the Android market provides the facilities for developers and users to submit and download the apps freely or for a price.

A key difference between the Android market and the Apple App Store is that, the Android market is open, whereas the Apple App Store is gated. That is, developers self-publish to the Android Market, whereas developers must submit applications for publication to Apple, and Apple decides what gets published [2].

In Android, applications must request permission at install time for any sensitive privileges they wish to exercise. Such privileges include access to the Internet, access to course or fine location information, or even access to see what other apps are installed on the phone [2]. Consequently, the properties that give Android apps the potential to improve our lives, also pose a serious threat to enterprise privacy and security.

Typically, Android users would click through terms of service and warnings, and therefore, unlikely to pay much attention to notices about Android permissions. Users tend to be unaware of the privacy impacts of their decisions. The disconnect between the user accepting security and privacy settings during installation and the enforcement of these settings upon subsequent application execution, along with ambiguous permission descriptions can lead to a fundamentally unaware user base [3].

Generally, there are two steps in obtaining permissions. First, an application developer declares that his or her application requires certain permissions in a file that is packaged with the application. Second, the user must approve the permissions requested before installation. Each application has its own set of

permissions that reflects its functionalities and requirements. Users can weigh the permission against their trust of the application and personal privacy concerns [4].

The official Android Market provides every application with two installation pages. The first installation page comprises a description, user reviews, screenshots and an "Install" button. After pressing "Install", the user is shown a final installation page that includes the application's requested permissions, as shown in Figure 1. Permissions are displayed as a three-layer warning: a large heading that states each permission's general category, a small label that describes the specific permissions, and a hidden details dialog. If a user clicks on a permission, the details dialog pops. The details dialog may include examples of how malicious applications can abuse the permission, e. g., "Allows an application to write to the USB storage". The permission system gives users a binary choice: the user can accept all of the permissions and proceed with installation.

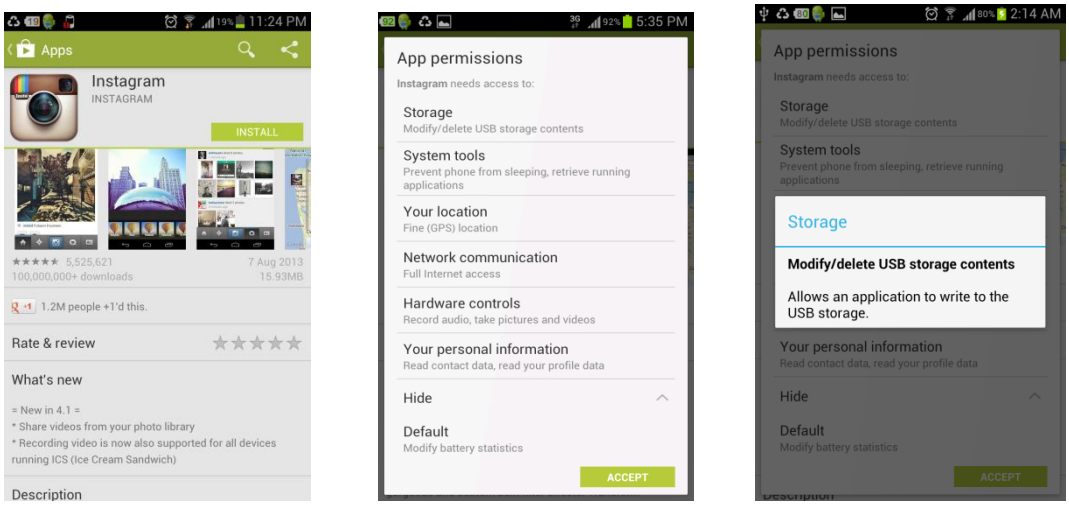


Figure 1. Top left, the Download page of an application, on the top right is the final installation page which displays the application's permission requests. On the bottom, the permission dialog that appears if a user clicks on a permission warning

Prior to this research, we have surveyed many literatures to find the state of research in this area. Many researchers have attempted to investigate various issues related to the Android permissions and data security and privacy.

In particular, Barrera et al. conducted and empirical analysis to investigate how the permission-based system in Android is used in practice by looking at whether the design expectations meet the real-world usage characteristics [5]. They introduced a methodology for exploring and empirically analyzing permission-based models using the Self-Organizing Map (SOM) algorithm.

In addition, Alhamed et al. perform a comparison on the privacy control methods implemented on the Android and iOS platforms. As a result, the propose a programming model for platform designers to improve privacy and they claim that using their proposed programming models, 14 Android and 5 iOS privacy bugs can be eliminated [6].

The work of Vidas et al. looks at the way permissions are managed in Android. It states that because developers do not have an easy way to determine which of the 130 application permissions their application needs, they end up specifying more permission that they need. This results in the violations of least privileges principle [3].

Stevens et al. [7] analyze about 10,000 free apps from popular Android markets and found that a significant sub-linear relationship between the popularity of a permission and the number of times when it is misused. Moreover, they also study the effect of the influence of permission (the functionality that it controls) and the interference of permission (the number of other permissions that influence the same classes) on the occurrence of both permission misuses.

Jeon et al. [8] investigate fine-grained versions of five of the most common Android permissions, including access to the Internet, user contacts and system settings. On top of that, they develop a suite of tools that allow the fine-grained permissions to be inferred on existing apps; to be enforced by developers on their own apps; and to be retrofitted by users on existing apps.

From user privacy point of view, Felt et al. [4] examine whether the Android permission system is effective at warning users. In the study, they evaluate whether Android users pay attention to, understand, and act on permission information during installation. They performed two usability studies: an Internet survey of 308 Android users, and a laboratory study where they interviewed and observed 25 Android users. From the studies, they found that current Android permission warnings do not help most users make correct security decisions.

Felt et al. and Chia et al. surveyed Android applications and identified the most-requested permissions [4], [9]. Chia et al. also found several correlations between the number of permissions and other factors: a weak positive correlation with the number of installs, a weak positive correlation with the average rating, a positive correlation with the availability of a developer website, and a negative correlation with the number of applications published by the same developer.

While other works look at various aspects in Android permission issues, we are interested at studying the factors that would influence the number of permissions in Android apps.

The main focus of this paper is to investigate the relationships between several variables, i.e., Number of downloads Price, and Rating with the Permissions in Android.

2. ANDROID PERMISSION MODEL

Android apps are programmed in Java, and compiled into Dalvik bytecode, and then packaged as a *.apk* file (APK), which is similar to Java's JAR file format. An APK consists of the app's class files; static assets such as image or video data, and a Manifest file [7]. The Manifest is an XML file that specifies a number of properties about the app, such as what Android API level the app is targeted for, what screen or Activity the app should start on, and what permissions the app requests.

Permissions are the basis for the Android security model; they are granted to the app for its lifetime while installed on the device, with the exception of updates which may change the app's permissions. The appropriate permission must be requested in order for an app to access sensitive device functionality, such as the network or GPS manager. Usually, each permission controls some set of Android or Java API functions, and calling one of these privileged functions without the appropriate permission will cause a security exception [10].

When a user installs an app, usually done through an Android app marketplace, all permissions requested by the app are presented to the user prior to installation, and the user is given an all-or-nothing choice of granting all the permissions and installing the app, or cancelling the install altogether.

A basic Android application has no permissions associated with it by default, meaning it cannot do anything that would adversely impact the user experience or any data on the device. To make use of protected features of the device, the developer must include in the *AndroidManifest.xml* one or more <uses-permission> tags declaring the permissions that your application needs.

At application install time, permissions requested by the application are granted to it by the package installer, based on checks against the signatures of the applications declaring those permissions and/or interaction with the user. *No* checks with the user are done while an application is running: it either was granted a particular permission when installed, and can use that feature as desired, or the permission was not granted and any attempt to use the feature will fail without prompting the user.

Often time's permission failure will result in a SecurityException being thrown back to the application. However, this is not guaranteed to occur everywhere. For example, the sendBroadcast (Intent) method checks permissions as data is being delivered to each receiver, after the method call has returned, so you will not receive an exception if there are permission failures. In almost all cases, however, a permission failure will be printed to the system log.

The permissions provided by the Android system can be found at Manifest.permission. Any application may also define and enforce its own permissions, so this is not a comprehensive list of all possible permissions.

3. SECURITY SENSITIVE DATA ON ANDROID

Each Android device contains a wealth of security-sensitive information associated with the user's identification, whereabouts, contacts, and many more.

A recent work by Wei et al. [2] reported various sensitive-security information which could potentially be manipulated by the Android apps, such as:

1. IMEI, IMSI, and phone number. The IMEI (International Mobile Equipment Identity), IMSI (International Mobile Subscriber Identity), and phone number are unique numbers that identify the physical device and the user's mobile subscription, respectively.

2. Contacts list. This list identifies all contacts which may include names, phone numbers, addresses, and emails.
3. Location. This information, provided by the GPS and the Network Location Provider, allows apps to determine the user's geographical location.
4. SMS messages. These include the private text, picture, and sound messages contained in the phone's message Inbox.
5. External SD card. The SD card (external storage) may contain a variety of personal files including photos, music, and documents. The card may also contain application-specific data.
6. Physical sensors. These sensors include the accelerometer, compass, light sensor, pressure sensor, proximity sensor, and temperature sensor. The sensor data is generated by hardware components directly measuring changes in the physical properties in the environment of Android devices.

In a similar line, Jeon et al. [8] developed a taxonomy that partitions Android permissions into broad categories based on the kind of resource being protected. Essentially, they categorized the permissions to four categories:

1. Category 1: Access to outside resources:

The first category contains 11 Android permissions that enable access to external resources, including Internet access, sending and receiving text messages, and reading and writing external storage. These permissions are naturally parameterized by the particular external resource being accessed. For example, Internet access involves specifying an Internet domain, text messages are sent to a phone number in a certain area code, and SD card access involves specifying a directory or file name.

2. Category 2: Access to Structured User Information:

The second category contains 14 Android permissions that access structured user data, such as a user's calendar, contact list, and account information.

3. Category 3: Access to Sensors:

The third category contains 7 Android permissions that protect access to various sensors on the phone, including the camera, GPS receiver, and microphone.

4. Category 4: Access to System State and Settings:

The fourth category contains 15 Android permissions that provide access to state and settings information on the phone. Typically each such permission provides access to read or update several unrelated pieces of information. For example, Android's READ PHONE STATE, among other things, allows an application to determine if a phone call is occurring and read the phone's unique IMEI number. Similarly, the WRITE SETTINGS permission allows an app to update many distinct phone settings, including the default ringtone and network preferences.

The taxonomy of threats to the Android Framework was discussed in [11], and they are:

1. 1.Abuse of costly services and functions (for example, sending short message service/multimedia messaging service (SMS/MMS) messages, making phone calls, or redirecting phone calls to high-rate numbers) by remotely exploiting a vulnerability in a core component that's exposed on the Internet.
2. Malicious activity against a network or network device (for example, sending spam, infecting other devices, sniffing, or scanning) by remotely exploiting a vulnerability in a core component that's exposed on the Internet.
3. Abuse of costly services and functions (such as sending SMS/ MMS messages, making phone calls, or redirecting phone calls to high-rate numbers) by an application exploiting vulnerability in a core component.
4. Malicious activity against a network or network device (for example, sending spam, infecting other devices, sniffing, or scanning) by an application exploiting a vulnerability in a core component.
5. Abuse of costly services and functions (such as sending SMS/ MMS messages, making phone calls, or redirecting phone calls to high-rate numbers) by maliciously using the permissions granted by the owner at installation.
6. Malicious activity against a network or network device (for example, sending spam, infecting other devices, sniffing, or scanning) by maliciously using the permissions granted by the owner at installation.
7. Disabling applications or the device by remotely exploiting vulnerability in a core component that is exposed on the Internet.
8. Disabling applications or the device by an application exploiting vulnerability in a core component.
9. Disabling applications or the device by maliciously using the permissions granted by the owner at installation.

10. Corrupting or modifying private content, or blocking, modifying, or eavesdropping on the device's communication network (for example, phone calls, Internet communication, emails, or SMS/MMS messages) by remotely exploiting a vulnerability in a core component that's exposed on the Internet.
11. Corrupting or modifying private content, or blocking, modifying, or eavesdropping on the device's communication network by an application exploiting vulnerability in a core component.
12. Corrupting or modifying private content, or blocking, modifying, or eavesdropping on the device's communication network by maliciously using the permissions granted by the owner at installation.
13. Obtaining, corrupting, or modifying private content when browsing a malicious Web site.
14. Blocking, modifying, or eavesdropping on the device's communication network when connected to an unreliable network.
15. Receiving spam, SMS/MMS messages, or emails.
16. Pushing advertisements to the browser application when browsing the Internet.
17. Loss of hardware components.
18. Causing a malfunction in hardware components.

4. RESEARCH METHOD

4.1. Research Questions

Based on the objective mentioned in Section 1, the following research questions were formulated:

1. What are the variables that have influence on the number of permissions in Android apps?
2. Which permissions are the most popular in Android apps?
3. Are the variables correlated with one another?

4.2. Research Hypotheses

The hypotheses in this research are:

H1: The number of permissions in Android is correlated with Price.
H2: The number of permissions in Android is correlated with Downloads.
H3: The number of permissions in Android is correlated with Rating.
These hypotheses will be tested using the data obtained from the App store repository.

4.3. Data Collection

For the purpose of data collection, we developed a data mining tool called OSSGrab [12], which can search through the Google Play Store and display the results in both HTML and Excel files. In Figure. 2, we can see that the Search options are divided into 2, the Simple Search and Advanced Search. The Simple Search will search for specific app, e.g. Instagram, while the Advanced Search option will look for the apps that match the criteria given by the user.

The example of the result in HTML format is shown in Figure. 3. The result shows the list of systems with several variables like Rating, Category, Downloads, Price and Permissions. These variables are important to be analyzed in this research.

We collected 5000 Android apps from the Android Play Store using the OSSGrab tool and analyzed them using a statistical package, SPSS. The results will be discussed in the next section.

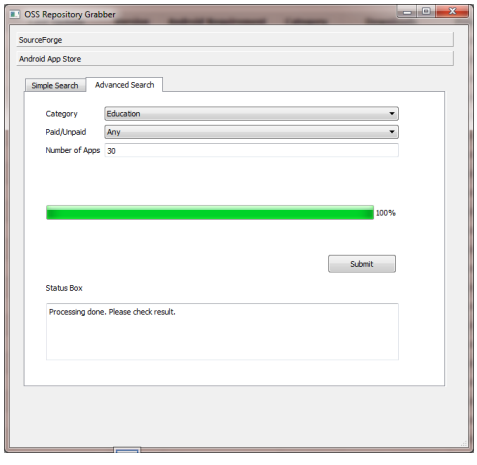
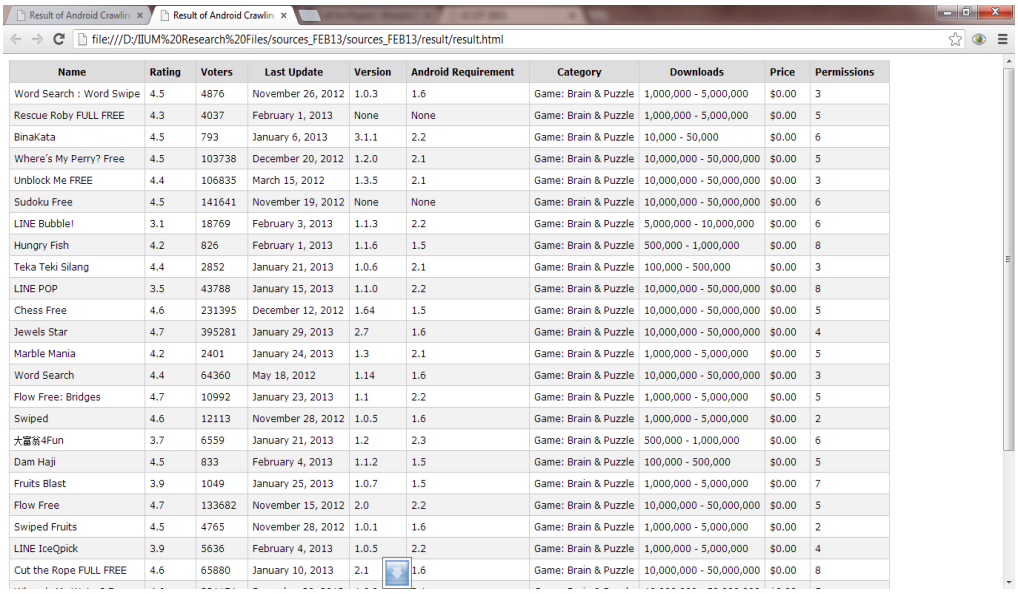


Figure 2. OSSGrab tool to extract Android apps from Google Play Store



Result of Android Crawlin | Result of Android Crawlin

file:///D:/IIUM%20Research%20Files/sources_FEB13/sources_FEB13/result/result.html

Name	Rating	Voters	Last Update	Version	Android Requirement	Category	Downloads	Price	Permissions
Word Search : Word Swipe	4.5	4876	November 26, 2012	1.0.3	1.6	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	3
Rescue Roby FULL FREE	4.3	4037	February 1, 2013	None	None	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	5
BinaKata	4.5	793	January 6, 2013	3.1.1	2.2	Game: Brain & Puzzle	10,000 - 50,000	$0.00	6
Where's My Perry? Free	4.5	103738	December 20, 2012	1.2.0	2.1	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	5
Unblock Me FREE	4.4	106835	March 15, 2012	1.3.5	2.1	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	3
Sudoku Free	4.5	141641	November 19, 2012	None	None	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	6
LINE Bubble!	3.1	18769	February 3, 2013	1.1.3	2.2	Game: Brain & Puzzle	5,000,000 - 10,000,000	$0.00	6
Hungry Fish	4.2	826	February 1, 2013	1.1.6	1.5	Game: Brain & Puzzle	500,000 - 1,000,000	$0.00	8
Teka Teki Silang	4.4	2852	January 21, 2013	1.0.6	2.1	Game: Brain & Puzzle	100,000 - 500,000	$0.00	3
LINE POP	3.5	43788	January 15, 2013	1.1.0	2.2	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	8
Chess Free	4.6	231395	December 12, 2012	1.64	1.5	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	5
Jewels Star	4.7	395281	January 29, 2013	2.7	1.6	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	4
Marble Mania	4.2	2401	January 24, 2013	1.3	2.1	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	5
Word Search	4.4	64360	May 18, 2012	1.14	1.6	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	3
Flow Free: Bridges	4.7	10992	January 23, 2013	1.1	2.2	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	5
Swiped	4.6	12113	November 28, 2012	1.0.5	1.6	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	2
大富翁4Fun	3.7	6559	January 21, 2013	1.2	2.3	Game: Brain & Puzzle	500,000 - 1,000,000	$0.00	6
Dam Haji	4.5	833	February 4, 2013	1.1.2	1.5	Game: Brain & Puzzle	100,000 - 500,000	$0.00	5
Fruits Blast	3.9	1049	January 25, 2013	1.0.7	1.5	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	7
Flow Free	4.7	133682	November 15, 2012	2.0	2.2	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	5
Swiped Fruits	4.5	4765	November 28, 2012	1.0.1	1.6	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	2
LINE IceQpick	3.9	5636	February 4, 2013	1.0.5	2.2	Game: Brain & Puzzle	1,000,000 - 5,000,000	$0.00	4
Cut the Rope FULL FREE	4.6	65880	January 10, 2013	2.1	1.6	Game: Brain & Puzzle	10,000,000 - 50,000,000	$0.00	8

Figure 3. OSSGrab search results in HTML format

5. RESULTS AND ANALYSIS

In order to answer the research questions and test the hypotheses, several analyses were conducted on the data. First, we conducted a normality test on the data and the results show that the distributions of the data are not normal.

Therefore, for correlation analysis, Spearman correlation analysis was done. Due to the non-normality of the data, a few variables were transformed into log base 10 values. For instance, Price is transformed into log Price, Permissions variable is transformed into log Permissions and lastly, Number of Downloads is transformed into logDownloads.

The overall results of the Spearman analysis are shown in Table 1. The abbreviations used in the table are defined as follows: Perm. Represents log Permissions, Pri. Represents log Price, down. Represents logDownload, and Rate represents Rating.

Table 1. Spearman Correlation between Variables

Metrics	Perm	Pri	Down	Rate
Perm	1	0.134 (0.000)	0.237 (0.000)	-0.129 (0.000)
Pri	-	1	0.02 (0.307)	-0.018 (0.395)
Down	-	-	1	--0.017 (0.234)
Rate	-	-	-	1

In Table 1, the top row represents the values for the Spearman correlation coefficient between the two variables, while the bottom row (in parenthesis) represents the p-value for the correlation. The results show that all the tested variables, i. e. Rating (Spearman corr. = -0.129, p-value = 0.000), Number of Downloads (Spearman corr. = 0.237, p-value = 0.000) and Price (Spearman corr. = 0.134, p-value = 0.000) are correlated with Permissions. Thus, our hypotheses H1, H2 and H3 are supported by the findings.

The correlation between Permissions and Price is depicted in Figure 4. Figure 5 illustrates the correlation between Permissions and Number of Downloads while Figure 6 exhibits the correlation between Permissions and Rating.

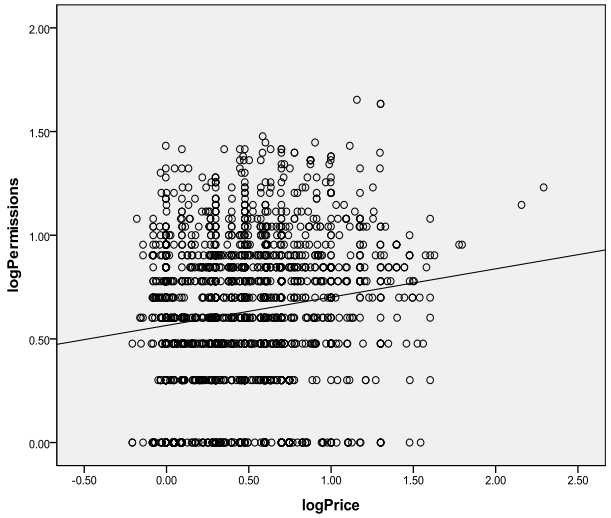


Figure 4. Permissions vs. Price

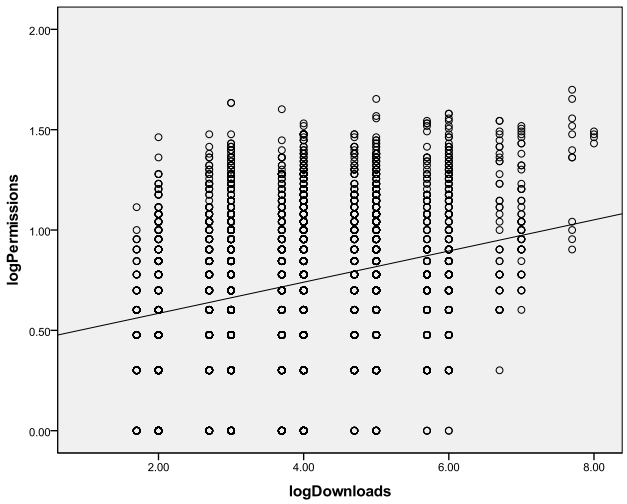


Figure 5. Permissions vs. Downloads

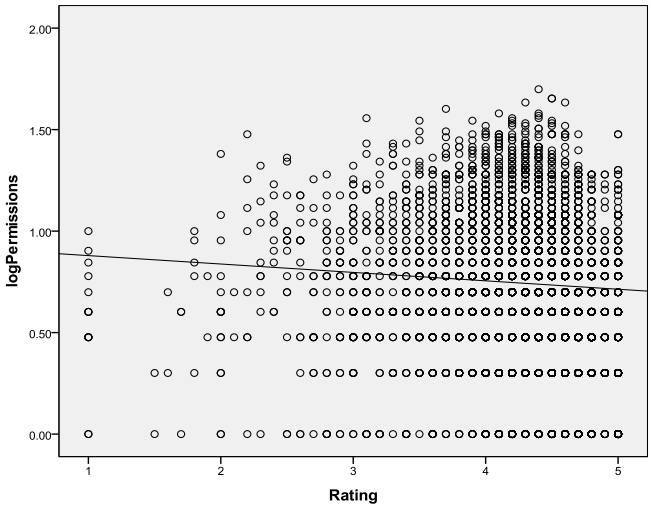


Figure 6. Permissions vs. Rating

In addition, we also did an analysis by system category and our data were collected from 8 major categories In the Google Play Store. Figure 7 shows the Mean value of Permissions and the results show that the Media And Video category has the highest average of permissions (Mean = 16), and the Tools category has the Lowest average of permissions (Mean = 4).

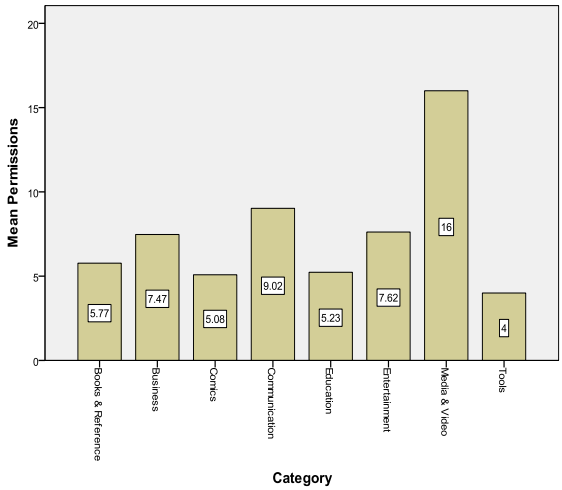


Figure 7. Permissions by System Category

Out of the 5000 Android apps we collected, 50% or 2503 apps are free apps and another 50% or 2497 are paid apps. A fine-grain investigation into the permissions shows that 6% of the free apps have zero permission while 14% of the paid apps do not have permissions. Figure 8 shows the permissions by price category and the free apps are represented by the label "1.00" and the colour blue. The Paid apps are represented by the label "2" and the colour green. The figure also shows that, in most cases, the free apps have more permissions than the Paid apps.

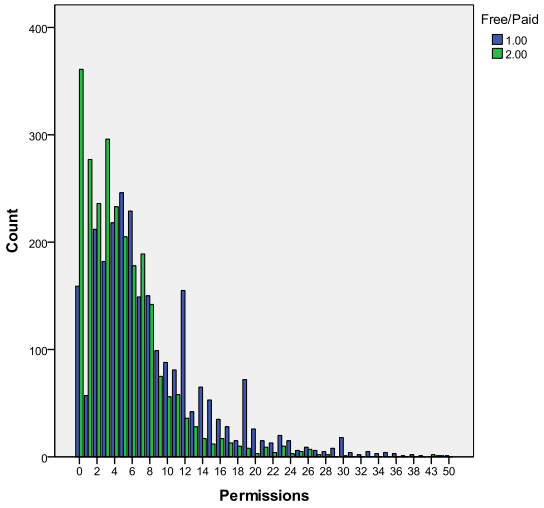


Figure 8. Permissions by Price Category

From the data we collected, there are a total of 101 different types of permissions being requested by the apps, and the number of permissions varies in each app. For example, a popular photo sharing app,

Instagram, requests nine permissions to be accepted by the users, which includes permissions to control USB storage, system tools, location, full internet access, hardware controls, user's personal information, modify battery statistics, receive data from Internet and discover user's accounts. If the users want to install the application, the users have to accept all permissions; there is no tick box on the choices of permissions which the users can select.

From the 5000 Android apps which we collected from the Android Play Store, we wish to highlight the top ten permission types. Table 2 shows the permission types and the respective number of systems which contain the permissions.

Table2. Top Ten Permission Types

Permission Type	Number of Systems
Full network access	3955
View network connections	3167
Modify or delete the contents of USB storage	2928
Test access to protected storage	2912
Read phone status and identity	2005
Prevent device from sleeping	1371
Control vibration	1290
View Wi-Fi connections	1218
Run at startup	825
Google Play license check	818

6. CONCLUSIONS AND FUTURE WORK

This paper applies the empirical software engineering analysis on the permissions in Android apps with several variables such as, Price, Number of Downloads and Rating. The findings indicate that all the variables have correlation with Android permissions, which means they have influence to some degree on the permissions. We also highlighted the top ten most used permissions in mobile apps, some permission are necessary but most of the permissions invade user's privacy, especially if they involve modifying user's data and reading sensitive data. Moreover, we found that out of the 5000 apps being investigated, 6% of the free apps and 14% of the paid apps have zero permission.

In the future, we plan to develop a tool to identify the apps that contain malicious permissions and provide users with the information on the threats which could potentially arise from the permissions.

REFERENCES

[1] Techland, Who's Winning, iOS or Android? All the Numbers, All in One Place. April 16 2013. http://techland.time.com/2013/04/16/ios-vs-android/#ixzz2XFRn70Rp

[2] X.Wei, L. Gomez, J. Neamtiu and M. Faloutsos, "*Malicious Android applications in the Enterprise: What do they do and how do we fix it?*" IEEE 28th International Conference on Data Engineering Workshops, pp. 251-254, 2012.

[3] T. Vidas, N. Christin and L. F. Cranor, *"Curbing Android permission creep"*, Proccedings of the 2011 Web 2.0 Security and Privacy Workshop. Oakland, California, May 2011.

[4] A. Felt, E. Ha, S. Egelman, A. Hanet, E. Chin and D. Wagner, "Android permissions: User attention, comprehension and behaviour", Technical Report No. UCB/EECS-2012-26, 2012. University of California at Berkeley.

[5] D. Barrera, H. G. Kayacik, P. C. Oorschot and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to Android", ACM CCS'10, Chicago, Illinois, October 2010.

[6] M. Alhamed, K. Amir, M.Omari and W. Le, *"Comparing privacy control methods for smartphone platforms"*, International Workshop on the Engineering of Mobile-Enabled Systems, San Francisco, California, May 2013.

[7] R. Stevens, J. Ganz, V. Filkov, P. Devanbu and H. Chen, "Asking for (and about) permissions used by Android apps", *Mining Software Repositories (MSR*) 2013, San Francisco, California, May 2013.

[8] J. Jeon, K.K. Micinski, J. A. Vaughn, A. Fogel, N. Reddy, J. Foster and T. Millstein,"Dr. Android and Mr. Hide: Fine-grained permissions in Android applications", SPSM'12, Raleigh, North Carolina, October 2012.

[9] P. H. Chia, Y. Yamamoto, and N. Asokan, "Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals," in Proceedings of the World Wide Web Conf., ser. WWW, 2012.

[10] Android Developer Documentation. URL: http://developer.android.com Accessed 10 September 2013.

[11] IEEE Magazine, Mobile Device Security, March/April 2010.

[12] N. S. Awang Abu Bakar and I. Mahmud, "OOSGrab: Software repositories and App Store mining tool", Lecture Notes on Software Engineering vol. 1, no. 3, pp. 219-223, 2013.

BIOGRAPHIES OF AUTHORS

Normi Sham Awang Abu Bakar is an assistant professor in the Department of Computer Science, International Islamic University Malaysia. She obtained her PhD in Computer Science at the Australian National University. Her research interests are in the area of empirical software engineering, open source quality, agile methodology, mining software repositories and software engineering education.

Iqram Mahmud is a final year undergraduate student majoring in Computer Science in KICT, International Islamic University Malaysia. He has been developing data mining tools in Python for research purposes since 2011. Iqram has been a member of U. Dhaka and IIUM teams to World Finals of ACM International Collegiate Programming Contest in 2009 and 2012 as a contestant.