VOL. 16, NO. 14, JULY 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

PERMISSION-BASED SECURITY GAPS IN ANDROID OS

Saqlain Abbas[1], Abdul Mateen[1], Saeed Ullah[1], Rubina Adnan[2] and Sahar Arshad[3]
[1]Department of Computer Science, Federal Urdu University of Arts, Sciences and Technology, Pakistan
[2]Department of Computer Science, COMSATS Institute of Information and Technology, Islamabad, Pakistan
[3]Department of Computer Science, Bahria University, Islamabad, Pakistan
E-Mail: saqlain_abbas1988@yahoo.com

ABSTRACT

Currently, mobile phones are one of the essential necessities that have everyday usage applications (apps). These are used for sending and receiving emails, watching pictures and movies, navigation, and much more. One of the most famous open-source mobile platforms is the Android which is installed on 79% of all new mobile devices. For the increase in usage of the Android operating system, security assurance of the user data is very important factor. The permission mechanism, required by the Android, requests for very sensitive permissions which are a matter of concern in most of the cases, but the user has to accept it to avail app installation. Now a day, researchers are focusing on the Android modification that includes the development of user-enhanced privacy over it. Many solutions are being proposed but with the flaw of reduced potential resulting from its modification. So it is required to develop such solutions where a user has the facility to review and edit permissions before the installation of an app. This paper provides a survey of permission-based security models and approaches for the Android operating system. The study also reveals and highlights the security issues of Android.

Keywords: Android, security, permission, application.

1. INTRODUCTION

In the modern era of wireless communication, everyone is familiar with the smartphone and computer devices. A smartphone is a device that offers almost the same functions as that of a personal computer along with the extra functionality of receiving and making calls. These devices can be used to read and send emails, newspapers, and present office presentations. Due to its mobility, smartphones are much ideal than computers. Mobile apps have increased the usage of smartphones (Sykes *et al*. 2015). The most important feature of the smartphone is its ability to run third-party apps. The software programs that are compiled to perform specific tasks are known as apps such as camera, calendar and contacts are normally pre-installed in smartphones while others need to be installed from different platforms.

A larger number of software firms as well as independent programmers are building apps for smartphones. According to Quattrone *et al*. (2015), the major issue that reduces the popularity of Android is its security. Modern techniques and platforms allow third-party developers to create applications and distribute them in the application-store or marketplace. The popular Application Stores are Google Play, Amazon App Store, and Go Apk. Negash, *et al*. (2015) discussed that Safety Risk Awareness has been found to provide Temporary Relief. According to Timur, *et al*. (2014) the first Android Trojan was found in 2010 which sends text message to certain numbers by itself. The Blackberry and iPhone (Apple) review permission of their stores. Permissions in Android-based apps are asked during the installation time of app. According to statista.com, most recently 2.8 million apps are reported to be found in Google play store up to march 2017. Mobile Apps such as Facebook, Messenger, WhatsApp, Twitter and Instagram get user permissions during installation. Their permissions include recording, taking pictures and videos at any time. Third-party developers can access a large amount of user data using standard API calls. Only Google play store gives a pop-up message about the required permissions for installing the app. The architecture for privacy and security varies among mobile platforms. Android depends on the software-based permission system which is shown in Figure-1.

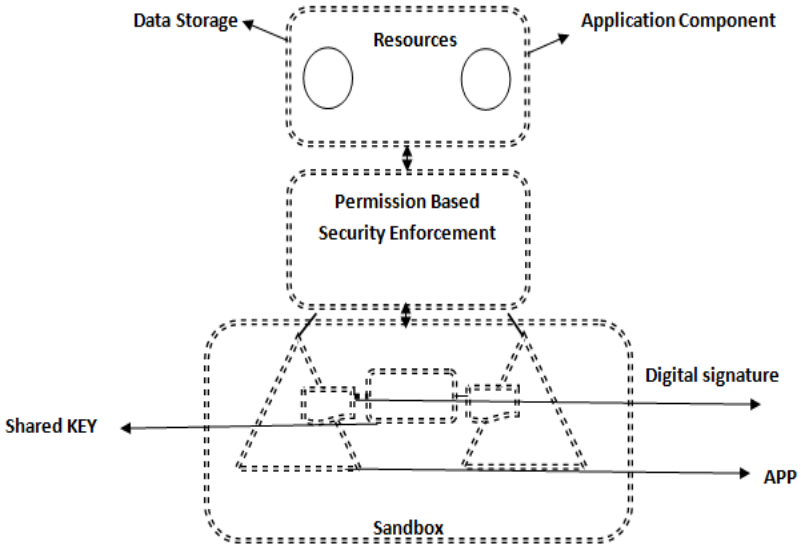

Figure-1. Android security model.

Apps installed on Android are controlled and manipulated in very odd way in respect of what impact do they have on system as well as on other installed apps. This controlled manipulation is called Sandboxing. So every app has its own sandbox with minimum involvement in other apps and has minimum effects on kind in other sandboxes. The shared key is used to secure the TLS/ SSL connection and mutually authenticate its peers while digital signatures are used for validation and integrity assurance of an App. Android Permission system is composed of three layers as shown in Figure-2.

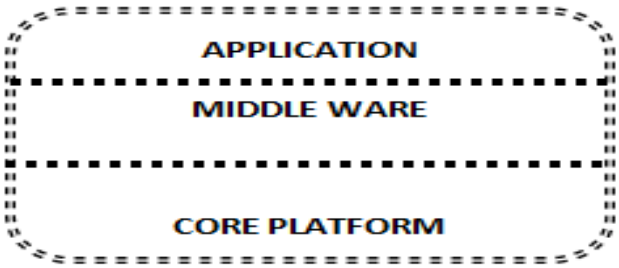

Figure-2. Android permission system.

The top Application layer is responsible to declare all permissions that are required to run an application which are granted by the users. The middleware layer provides an interface to interact with the resources in the core platform. This layer is also responsible to check whether permissions asked by the application are valid or invalid. The bottom layer is the core platform where all resources of the Android device are present. In order to access these resources, one should need permission from the user. According to techtimes.com (2017) some apps are only developed to collect and expose user personal information that may be used by cyber-criminals. There are cybercrimes, which take sensitive data including personal information, bank account information, and passwords in order to take their own benefits (Troels Fibæk Bertel 2013). Android apps developers have to write the needed permissions explicitly in a config file named Android.Manifest. In order to reduce security risk, Android official document has been launched. According to Lingfeng *et al*. (2016), there are 151 system-level permissions and 4000 classes in the Android library but even then this official Android documentation is incomplete.

The Rest of the paper is organized as follows: Section II describes the various studies and literature which are related to the security of the Android operating system. Section III is dedicated to highlight the security issues of Android. Finally, Section IV concludes the paper with some future directions.

2. LITERATURE REVIEW

The major aim of this study is Android's security because of its tremendous usage among people. Normally mobile users don't consider the requested permission from application store even they don't know the exact meaning of these permission requests. Some research studies that are related to the security models and architectures of smartphones are discussed in this scection:

Negash *et al*. (2015) conducted a survey on Android's permissions. It was focused on creating awareness about security threats to data. Their focus was on the impact of risk and trusting belief when someone has the intention to use or install any mobile App. In this survey, the participants are the students, having an age group of 26-39 with both male and female. In their proposed setup, they performed pre-test and post-test. In the pre-test, the participants used a normal calculator app whereas in the case of post-test embedded, visual questions and highlighting the negative impact of permissions were performed. The results of this test are mixed and need more investigation. However, their basic research results show that an increase in awareness about

permission reduces the trust level on app developers. In another study, Jain *et al*. (2016) created two Apps named "Contacts "and "Get data" by using the Eclipse platform. The "Contacts app", installed on the user's cellphone stores user contacts. It requires permission to access phone contacts and full internet access. "Get data" was installed on the hacker's cellphone. From Get data app, they fetched the user contacts by using contacts app and showed that how hackers can misuse the permissions. The aim of the research was to show the protection of a user on Android devices from the permission-based attacks, according to their experiment, they inferred that a valuable tool or software is required, that should control the permission before installation. Quattrone *et al*. (2015) proposed a method that uses app similarity to detect anomalies. They used isolation forest a data mining technique for detecting outliers to find anomalies. They focused on Android and developed a privacy palisade service. Web scrapper is used to collect information of 17000 apps from the Google Play Store. A service with the name privacy palisade is developed which checks apps permission on Android. If the applications have outliers then the user is notified and the user can see color-coded launcher icons for privacy. The dialogue window was opened whenever an outlier app was opened. The OS-level modification is done in this research. The privacy palisade is a light service that does not use many resources. To build a customized launcher, they downloaded the source code of Android 4.4 kit Kat and compiled on Linux. Bao *et al.* (2016) used an approach based on collaborative filtering. The proposed approach is evaluated on 936 open-source android apps which are taken from F-Droid having corresponding GitHub repositories. The approach is tested in terms of precisions. To extract API from Java Source code the source code is converted into a single XML file using SrcML which is a lightweight static analysis tool. By using XML processing tool permissions are extracted and APIs from app are then compared with other similar apps. The Limitation of their work is merely focus on F-Driod and do not take a single app from other Stores. Kraus *et al*. (2014) provided the solution about the permission requested by different applications in the form of statistical analysis that provides extra information about these requests. Results from this study have created awareness among users that they should choose applications with lesser permission steps. But this study also has certain limitations such as it is almost impossible to evaluate such types of statistical analysis for newly launched applications as this study gives satisfactory results for apps with alternatives present. Bartel *et al*. (2012) presented a basic approach to minimize the hackers' attack. A simple matrix is used to detect Apps having malicious permissions on Android. The future work of the researcher is to discover how to express permission implementation as a crosscutting concern, in order to automatically add or eliminate permissions enforcement. Fauzia *et al*. (2014) proposed a permission and intent model together for malware detection. The researchers also proposed a new concept of benware application which is same as benign but has unnecessary permissions that are not required for their functionality. The classification algorithms such as were naïve Bayes, KStar and Prism are used to validate the experiments. Quang *et al*. (2014) investigated the permissions of the application by manual method. The primary benefit of this method is that any app can be installed on any version of Android which supports the unmodified app. In this approach, there is no root required for the removal of permissions. In this method, the app is first decompiled using APK Multi tool where smali code files are presented but not in readable format. These files are then converted to dex or delvik executable file using a tool named as smali/ baksmali. After converting, changes to permission are made and then the app is converted back into APK. Kaur *et al*. (2014) introduced a permission modification approach through which a user can modify permissions being requested by the application. Using Pemo, a user can stop and resume the permission from the application. Pemo works with the assistance of the Package Manager that is used for installation, up-gradation, and removal of an application. The permission check system is modified through which they first check the permission of the app. If the app has more permission then it is uninstalled by the framework so it can be installed with modification in permissions. Their Framework is evaluated on the true caller, offline dictionary, book my show, 2048 Game, Paper Toss, Flashlight, Pics Art and Color Note. The proposed framework is failed on Pics Art because it stopped working after removal of permission and an error was shown in Offline Dictionary. To spot out Android malware at a very initial level, one of the apps named Riskranker was designed by Michael *et al*. (2012). This app tries to access potentially harmful risks imposed by mistrusted apps. an automated system that could energetically analyze the risks of app. A two step risk analysis was the base of assessment. Initially it detects apps with low and average risk of malware security issues. In order to diagnose such apps, it identifies uncomplicated performances of apps that appeals (i) initiation of root activities (ii) misleading cost creation (iii) security risks to user's privacy. In the second step of analysis, apps that encodes the codes for various activities that have bypassed the first step of analysis are further being analyzed for doubtful behavior. Apps with these and related issues are being mapped by a set of instructions developed by scientists. In order to access the future and recommended solution, they developed a model using 118,318 apps (104,874 distinguishing apps) taken from different legally official and unofficial stores. After this proposed assessment, 1st step has diagnosed 2,461 strange apps and 2nd step has located 840 apps. Among these 3, 281 apps, the main success was uncovering of 322 accounting to 9.81% (of total located apps) that belongs to 11 different families. Compatibility of permission request taken from apps is measured by a solution, called as WHYPER, (known as Natural Language Processing) proposed by Pandita *et al*. (2013). The apps related details as given by developers and the need why permission request is mandatory are used as a criterion for analyzing compatability. Application details provided by developer

and a semantic model are used as an input, and are used to find which thing in apps details defines the exact use of permission. WHYPER was tested using 581 apps taken from famous Android app stores. Results have shown 82.8% precision and 81.5% recall for three important permission (address book, calendar, and audio record) that are normally considered to protect user's privacy issues. The main challenge to WHYPER is those apps that have no description provided by the developer that causes a false-positive diagnostic. One of the proposed solutions to diagnose Application communication liabilities was ComDroid which was proposed by Erika *et al*. (2011). The majority of these liabilities originate from the details that aim for intra and inter-application communication. ComDroid on one-hand analyses interactions among applications and on the other hand leads to the diagnosis of security risks imposed by these interactions. Security risks include leakage of private information, exploitation, and many unpredicted actions. ComDroid is a two-step solution, in the first step, un-strapping apps DEX files by use of common man accessible tool Dedexer. The second phase includes analysis of the unstrapped outputs from Dedexer and potential security threats. An analysis of 20 apps, done by ComDroid has shown 34 potentially harmful features contained in these apps, and 12 out of 20 apps have at least one of these threats. Beresford *et al*. (2011) developed an app through which they saved the sensitive data by giving fake permissions and allowed the user to provide fake or mock data to application interactively as the application is being used. Using mockdroid, the user can cancel permission to particular resources at run-time and encourages them to use functionality of app without disclosing their sensitive data .for e.g. providing "no location fix" when it is being asked the GPS location by a specific app. This could be one of the ways to protect leakage of personal data. "App Opps "was originally introduced by Google in Android 4.3 as a hidden feature. With the release of kit Kat, Google made it difficult to access "app ops" but it was still possible to access it with rooting. The main limitation of this app was that it only blocks access to those things which are willing to be blocked. It does not provide any ability to control whether an application should have right to use to internet. Google can push its advertisement and gather user data by app ops. It is recommended to use other alternatives than using the App Ops. The summary of the above discussed approaches and framework is shown in Table-1.

Table-1. Analysis table of the literature review.

Reference	Summary	Tools	Algorithm/ Methodology
Quattrone *et al*. (2015)	Privacy palisade method is adopted to check apps permission on Android	-	Isolation Forest Algorithm
Negash *et al* (2015)	A survey was conducted to check awareness about android Permissions awareness	Questionnaire	Survey through Questionnaire
Bao *et al* (2016)	A comparison with other apps is performed by extracting information of permissions	SrcML & XML	Collaborative Filtering
Quang Do, *et al* (2014)	The reverse engineering process is applied to remove an app's permission to a resource	APK Multi Tool Virtuous Ten Studio	Reverse Engineering
Jain *et al* (2016)	By installing malicious app, user can lose his private and confidential data	Eclipse platform	-
Kaur et. al (2014)	The proposed framework is used to modify the apps permission	Android Eclipse Emulator	-
Kraus *et al* (2014)	Statistical information about permissions is gathered through functionality	Fisher's exact test	user study
Fauzia *et al* (2014)	New concept of benware is Proposed and the proposed work is verified by using classification algorithms	-	Classification algorithms
Bartel *et al* (2012)	Method to decrease the attack surface of permission-based software is presented	-	Static Analysis Method
xda-developers	App Ops deployed by Google to control permissions	-	-
Grace *et al* (2012)	Riskranker was designed to spot out android malware at very initial level	-	-
Pandita *et al* (2013)	Compatibility of permission request taken from apps is measured	NLP	-
Erika *et al* (2011)	Solution to diagnose Application communication liabilities	Dedexer	-
Beresford *et al*. (2011)	The aim is avoiding of giving out sensitive data by granting fake permissions	-	Package Manager modifications

3. DISCUSSION AND ANALYSIS

The above discussion shows that a number of research studies have been conducted to increase the awareness about permission issues of the applications. Negash, *et al* (2015) proposed that adopting a demo app with visual questions can increase the awareness effects and user reaction during the installation of apps. Yet, the results are mixed but their basic aim was to provide the awareness of the user about the permissions of application which is to be installed on Android-based mobile phones. The limitation of this survey is that the sample size of the data is very small. Bartel *et al*. (2012) used a tool on a dataset of Android apps and identified that a non-negligible part of applications suffers from permission gaps. In another study that is performed by Fauzia *et al*. (2014). A new concept of benware application is introduced and this is first ever malware detection strategy which classifies applications into three sorts (malware, benware and benign). The idea is verified through classification algorithms which provided very positive results. In order to remove or edit permissions of installed application Quang *et al.* (2014) manually altered the permissions of the application which is time-consuming and tedious. Moreover, most of the applications are digitally signed and after editing the apps cannot be updated. A permission modification approach is developed by Kaur *et al*. (2014) through which applications permissions are modified but again technique is applied on limited applications and their tool needs Rooting of Android. Riskranker was developed by Michael *et al.* (2012) but it is worthy to mention that the major challenge of Riskranker is that they used to have the same set of instructions for coding and decoding, which is not effective in any respect. But from this research work, it is evident that applications gather more data than they have to work and send it to outer sources. It is just simple to get such data on versatile stages. It is an open inquiry on where the line ought to be drawn on such data breaks, and how much applications need to uncover to the client about how their data is being utilized. Pandita *et al*. (2013) proposed WHYPER, a framework that utilizes Natural Language Processing (NLP) strategy to decide why an application utilizes authorization. There are a few consents that are certainly utilized by applications and along these lines will have poor outcomes with WHYPER. According to the researchers, they don't anticipate that the Internet permission will function admirably with WHYPER. As all cellphone applications usually get connected to the Internet, and expect that activities will make a semantic chart for the Internet permission will be to a great extent inadequate. Comdriod a tool by Erika *et al*. (2011) depends on DEX code, so developers or analysts for the Android Market can utilize it to assess applications whose source code is unavailable. It is critical to take note that ComDroid does not confirm the presence of attack.

4. CONCLUSIONS AND FUTURE WORK

The basic aim of this study was to investigate the permission-based attacks targeting Android-based devices. The investigation of Android apps shows how extra permission can be used to steal user's private and personal data. A lot of work has been done to improve the user's interface so that the user can pay attention towards the permissions and can also understand the meaning of requested permissions. But even then, most of the people do not read the details of requested permissions and just allow it. There are lots of over-privileged apps which are used by the hackers through which they can perform unfair, steal and harm user devices. So there is a need to develop such security framework through which a user can control particular permissions. By adopting these steps, the user can edit or remove application permissions before the installation of any app. Therefore, we aim to develop a security model that is more secure for smart devices in the future.

REFERENCES

Bao L., Lo D., Xia X. & Li S. 2016, November. What Permissions Should This Android App Request?. In Software Analysis, Testing and Evolution (SATE), International Conference on (pp. 36-41). IEEE.

Bartel A., Klein J., Le Traon Y. & Monperrus M. 2012, September. Automatically securing permission-based software by reducing the attack surface: An application to android. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (pp. 274-277). ACM.

Bertel T. F. 2013. It's like I trust it so much that I don't really check where it is I'm going before I leave. Informational uses of smartphones among Danish youth. Mobile Media & Communication. 1(3): 299-313.

Beresford A. R., Rice A., Skehin N. & Soha R. 2011, Marc. Mockdroid: trading privacy for application functionality on smartphones. In Proceedings of the 12th workshop on mobile computing systems and applications (pp. 49-54). ACM.

Chin E., Felt A. P., Greenwood K. & Wagner D. 2011, June. Analyzing inter-application communication in Android. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252). ACM.

Do Q., Martini B. & Choo K. K. R. 2014, January. Enhancing user privacy on Android mobile devices via permissions removal. In System Sciences (HICSS), 2014 47th Hawaii International Conference on (pp. 5070-5079). IEEE.

G. Paller. 2017. Dedexer. Online; accessed at June 26 2017, http://dedexer.sourceforge.net

Grace M., Zhou Y., Zhang Q., Zou S. & Jiang X. 2012, June. Riskranker: scalable and accurate zero-day android malware detection. In Proceedings of the 10th

international conference on Mobile systems, applications, and services (pp. 281-294). ACM.

https://www.xda-developers.com/protecting-your-privacy-app-ops-privacy-guard-and-xprivacy/
http://www.techtimes.com/articles/18762/20141026/flashlight-appsare-spying-on-users-android-ios-windows-phone-smartphones-isyours-on-the-list.htm Retrieved June 30, 2017.

Idrees F. & Rajarajan M. 2014, October. Investigating the android intents and permissions for malware detection. In Wireless and Mobile Computing, Networking and Communications (WiMob), 10th IEEE International Conference on (pp. 354-358). IEEE.

Jain A. 2016, March. Android security: Permission based attacks. In Computing for Sustainable Global Development (INDIACom), 2016 3rd IEEE International Conference. pp. 2754-2759.

Kaur A. & Upadhyay D. 2014, September. PeMo: Modifying application's permissions and preventing information stealing on smart phones. The Next Generation Information Technology Summit (Confluence), 2014 5th IEEE International Conference. pp. 905-910

Kraus L., Wechsung I. & Möller S. 2014, July. Using statistical information to communicate android permission risks to users. IEEE Socio-Technical Aspects in Security and Trust (STAST) Workshop, 2014 on (pp. 48-55).

Mirzoev T., Brannon M., Lasker S. & Miller M. 2014. Mobile Application Threats and Security. World Comput. Sci. Inf. Technol. J. 4(5): 57-61.

Negash S. & Shahriar H. 2015, December. Mobile app permissions awareness. IEEE 5th International Conference on Information & Communication Technology and Accessibility (ICTA), 2015 pp. 1-4.

2017. Number of available applications in the Google play store. http://www.statista.com/.Spying on users' smartphone.

Pandita R., Xiao X., Yang W., Enck W. & Xie T. 2013, August. WHYPER: Towards Automating Risk Assessment of Mobile Applications. In USENIX Security Symposium. pp. 527-542.

Quattrone A., Kulik L., Tanin E., Ramamohanarao K. & Gu T. 2015, December. PrivacyPalisade: Evaluating app permissions and building privacy into smartphones. IEEE Information, Communications and Signal Processing (ICICS), 2015 10th International Conference on pp. 1-5.

Sykes E. R., Pentland S. & Nardi S. 2015, November. Context-aware mobile apps using iBeacons: towards smarter interactions. In the Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering. pp. 120-129.