Android Permission System Analysis According to User Activities

Amit Kumar Jha, Asst.Prof in Raajdhani Engineering College, Bhubaneswar
Vidya Mohanty, Asst. Prof in Aryan institute of Engineering and Technology,Bhubaneswar
Rakhi Jha, Asst. Prof in NM Institute of Engineering and Technology,Bhubaneswar
Ashish Singh, Asst. Prof in Capital Engineering College, Bhubaneswar

Abstract: In today's world there has been an exponential growth among smart-phone users which has led to the unbridled growth of smart-phone apps available in Google play store, app store etc., In case of android application, there are many free applications for which the user need not shell out a penny to use the services. Here the magic word is "free" which entices millions of pliant people into installing those apps and giving unnecessary access to their data and device control. Current studies have shown that over 70% of the apps in market, request to gather data digressive to the most functions of apps that might cause seeping of personal data or inefficient use of mobile resources. Of late, couple of malignant applications gather unobtrusive information of the user through third-party applications by increasing their permissions to high-level on the Android Operating System. Android permission system provides, the user access to the third party apps and in return based on the permissions granted by the user, an app can access the related resource from the user's mobile. A user is bound to grant or deny permits during the installation of the application. For the most part, users don't focus on the asked permissions, or sometimes users do not understand the meaning of the permission and install the app on their device. They allow a way for attackers to perform the malicious task by demanding for more than expected set of permissions. These extra permissions permit the attacker to exploit the device and also retrieve sensitive information from it. In this research paper we describe how permission system security can create an awareness among the users that would assist them in deciding on permission grants. This improved and responsible user activities in Android OS can help the users in utilizing their device securely.

Introduction

Android is simply an operating system that facilitates a user to interact and manage mobile devices through a Graphical User Interface (GUI). Some of the features of an android driven smart-phone are GPS capability, camera functionality, internet accessibility, touch screen interface, provision for application installation which is the main differentiating and important feature in comparison with generic mobile phones. To run these applications, Android devices support Operating System (OS) in the similar way as computer supports the operating system. Some most popular OS are Windows, Android, Linux, iOS, etc.

Android is the widely used open source operating system. This in-turn makes it very difficult in managing as any developer can make application in their own way and user isn't aware of pitfalls in the application, about its background services and activities and the related security threats.

Android operating system is based on Linux kernel. There are four layers in Android architecture. Each layer has different tasks. The base layer is Linux kernel which maintains Android operating system security and other components. This layer contains all device drivers, USB drivers, Bluetooth drivers, Wi-Fi drivers, display drivers and it is also helpful in maintaining power management of Android system.

. After rooting the device hacker or attacker can have direct access to Linux kernel. Once the device is rooted, then there is no permission required for accessing the device.

Native libraries are the upper layer of Linux kernel
which mainly consist of all kind of default libraries, for example, SQLite is for all database related operation, Webkit is used for inbuilt web browser, OpenGL is utilized for 2D and 3D graphics in Android and SSL is giving network access related authentications. The developer can use all libraries in their application, but many times attackers may put some extra permission in their apps and might misuse those libraries.

Android runtime also provides the core libraries and most importantly Dalvik Virtual Machine (DVM) facilities which help to run the application on the device. DVM as compared to JVM optimizes the Android device providing fast performance while consuming less memory.

Android Framework provides us with the lot of APIs like package manager, View Manager, content provider, Activity Manager, Resource Manager and also provides lots of classes and interfaces for Android application development.

The existing android system is a permission based system. For most applications, there are a set of permissions that need to be accepted for successful launching of the application. Most of these permissions concern with user sensitive data that are not needed for that application, for example a gaming app asking for permission to access contacts. So, the proposed system lets the user know how many other users have either accepted or rejected these permissions.

Permission System

In Android, each application has one manifest .xml file and in the absence of this file, task of running the application is nearly impossible. It contains the entire list of activities which are used in the apps and additionally it also includes all the permissions which the application needs.

Android forces apps to declare the permissions during the installation. The app user has to decide to grant or revoke the permission of any android applications before or after the installation. Malicious apps cannot be a treat to device until user allow access to demanded permissions. Most of the time user allow application to access android sub-services unknowingly, which causes improper working of device. To create basic awareness, the user could decide whether to allow to access certain permission or not. By providing information at bottom of the permission box, about how many people liked or disliked the set of permissions asked by application at its first time of use. Based on number of likes it helps the user to make certain decisions. Providing this reference, it helps to reduce android threats, crime and also protect the sensitive data. Given below are some sorts of permissions mostly used in the android application:

- Android.permission.READ_CONTAC
- Android.permission.WRITE_CONTACT
- Android.permission.READ_STORAGE
- Android.permission.WRITE_STORAGE
- Android.permission.RECEIVE_SMS
- Android.permission.WRITE_SMS
- Android.permission.SEND_SMS
- Android.permission.READ_SMS
- Android.permission.INTERNET
- Android.permission.READ_PHONE_STATE

Through the above given set of permissions, the applications can have access over all the resources. This manifest file is written within the XML which contains some kind of tags. Through those tags one can define the usage of application and structure of permission.

In Fig. 1 we can see the architecture of the Access Permission in Android. This is a basic model and it's same for all android devices. There are two applications, Application 1 and Application 2 where application 2 provides access to local data like contacts, sms, etc. and device component control like camera, mic, etc. In Application 1, there is one module "A" and in Application 2 there are modules "B" and "C". Module A can access the module "B" and "C" if they are assigned permission labels of Application 1.

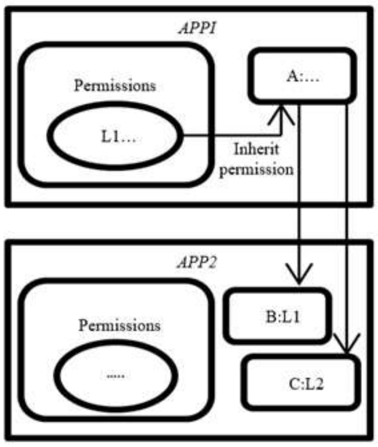
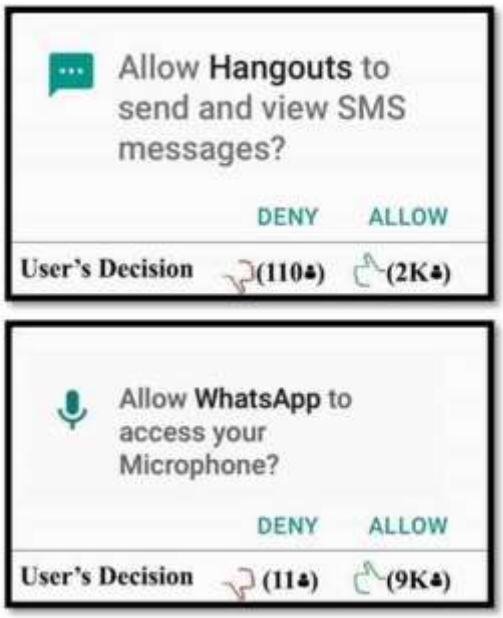


Fig. 1: Access permission architecture (Enck *et al.*, 2009)allowed or denied, but after applying this module users will frequently be able to decide whether the permission need to be allowed or denied.

In the module, we are putting one label with permission which is recommended to the user to allow or deny the permissions. This special label shows the how many users had allowed or denied that special permission. Based on working of application, its requirements must be fulfilled. If user does not grant permit to any one of the permissions, those service will not be accessed by application.. So blindly never grant access to any application to use our device services.



Allow **Hangouts** to send and view SMS messages?

DENY ALLOW

User's Decision **(110👤)** **(2K👤)**

Allow **WhatsApp** to access your Microphone?

DENY ALLOW

User's Decision **(11👤)** **(9K👤)**

Fig. 2: Demo view of permission module in Android

Implementation

When the user gets his device connected to the internet the Android local database syncs with server database given the application is installed on the device. All devices are recognized through a unique device id similar to the MAC address of PC and other computing devices. Through the given device id, we can get the information about installed application like the application ID. After getting application id, one can easily view all the permissions utilized by particular application and can also insert entries in a database.

In the database, four initial entries do appear Device id, Application id, Permission Label and State. State demonstrates that permission accepted or denied by the user. The value of the state is 0 or 1. Zero means Denied and One means Allowed appropriate permission.

Figure 4 shows the working of getting the permission data whether the user has denied or allowed the permission request of the application. When any application is installed by the user in their device, the application might ask for the set of permissions to access the device components like camera, microphone, storage or user's data like location, contacts, SMS etc. The applications have different types of permissions with different functionalities. In this algorithm I'm trying to get the data from the user's device to the server which is the choice of the user about to allow or deny the request of the permission.

In the server database the information which is to be stored is DeviceID, AppID, PermissionLabel, A = allow and D = deny. The local database information will be stored likewise. Depending on the device's internet connectivity if the device is connected to the internet it will store the data in both the databases at the same instance, otherwise it will store the data in the local database and will update the server when the internet will be connected.

When the user allows the permission, the allow variable (A) will be incremented by 1 and the whole record with full information like AppID, DeviceID, PermissionLabel, A = 1 and D = 0. When the user allows the permission, the deny variable (D) will be 0. Same will be vice versa when the user denies the permission (A = 0, D = 1). Once the choices are made by the user, it will check for the internet connectivity and if connected, the data will also be stored in the server database, otherwise only in the local database and will update in the server on establishing the connection.

Algorithms and Results of the Proposed System

There are two different algorithms that have been used in the proposed system. These algorithms are used to fetch the data from user's device and displays it. The algorithms and the results are as follows:

Algorithm 1: Get Data from user's Device

Now, the application permission data from the user device has been fetched using the above algorithm. For displaying how many users have allowed or how many has denied, the data will be displayed from the server with the different algorithm.

So, here we can see an example of the database and how data is stored. The database comprises of five columns. First is device id, which will store the all users device id. Second is App id, which will store the app id of install applications by the users and third is Permission, It stores permission asked by the installed app. The 3rd and 4th column is Allow and Deny. It will store the operation performed by the user on particular app permissions.

There a two ways or scenarios in which the database in the server gets updated.

During first time installation and running of the software, based on the customer preferences on the permission request, the device ID, App ID and different permissions and its status would be added as a new line item in the database.

During second time or repeated installation and running of the software, only the status of each permission request would be updated as per the user choice. A new line entry for the same device ID won't be created to avoid duplicate entries that can impacted.

In this example the chatApp has been installed on six devices. The chatApp has many permissions but here we mention only two namely "READ_CONTACT" and "BLUETOOTH" permissions. 1st, 2nd and 4th users allow "READ_CONTACTS" permission and 3rd user denies that permission. 5th user allows "BLUETOOTH" permission and 6th user denies that permission. This example shows how server database stores all data.

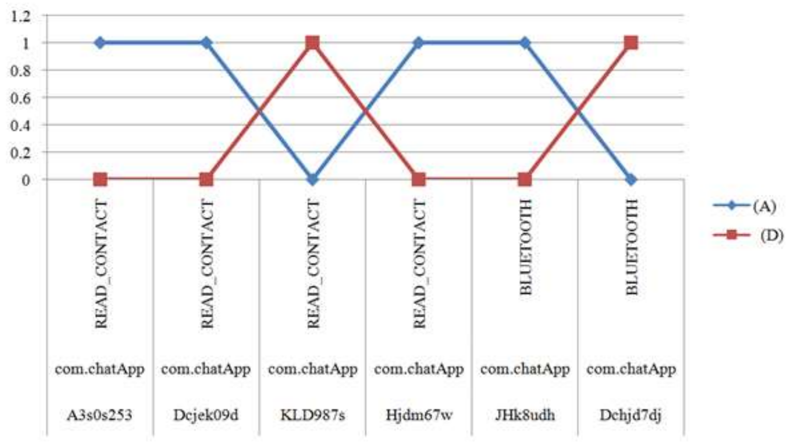


Fig. : Graph of Table 1

When a user initiates an application installation process from any App store, the OS initially checks whether the app is already installed in the device or not. If not, then it will immediately stop the process. Otherwise, during the installation process it will get the App ID and Device ID for that particular device and store it in the local database. After that it will fetch the count of Allowed (A) and Denied (D) based on the App ID and Device ID for the particular permission from the server to the device. This permission statistics data from the server helps the user in deciding which permission request he should honor or not as per the public opinion. Based on the choices made for each permission request, a line record would be created for each permission request with its status in the local device.

Algorithm 2: Fetch Data from Server to User's Device

Now after fetching the data from the server, it will check whether the App ID is there in the local database or not. If not, then data will be fetched from the server with information like Device ID, App ID, Permission, A = n, D = n (n = number of counts). But if found, then it will compare the permission and its global status count. If there is no difference then, it will stop as it contains all the data updated as in the server. In case, there are any changes in the two databases, it will update the local database with specific records with the data fetched from the server and will store the final content of data in the local database.

Table 1: Example of how data store in server

Device ID	App ID	Permission	(A)	(D)
A3s0s253	com.chatApp	READ_CONTACT	1	0
Dcjek09d	com.chatApp	READ_CONTACT	1	0
KLD987s	com.chatApp	READ_CONTACT	0	1
Hjdm67w	com.chatApp	READ_CONTACT	1	0
JHk8udh	com.chatApp	BLUETOOTH	1	0
Dchjd7dj	com.chatApp	BLUETOOTH	0	1

Table 2: Example of how data store in local database

Device ID	App ID	Permission	(A)	(D)
Abcde1	com.chatApp	READ_CONTACT	3	1
Abcde1	com.chatApp	BLUETOOTH	1	1

The following database table shows the information transferred from the server.

Now, when the user opens the application after installation, it will ask for certain permissions to the user. As shown in Fig. 2 it will show a dialog box with two options namely Allow and Deny for any particular permission like using Camera, accessing Contacts etc. It will additionally show the count of how many people allowed the permission for that particular application and how many denied for the same.

In Fig. 7 example server data is shown in Table 1, In that table total three users allowed "READ_CONTACT" permission and one user denied. For "BLUETOOTH" permission, one user allowed and 1 user denied that permission. Same count summary of appropriate permissions is shown in Table 2.

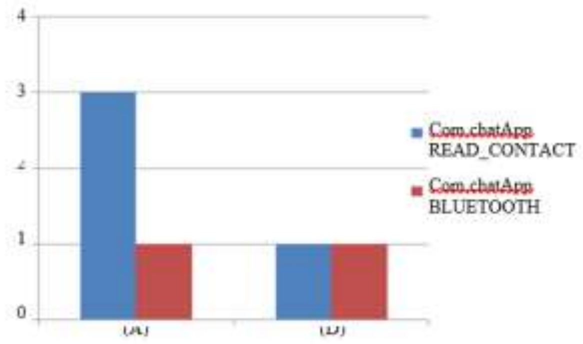


Fig. : Graph of Table

Results and Discussion

As we see in algorithm 1 the individual data of each user is fetched. The permission name along with its status are extracted for each user. This data has been processed by algorithm 2 to generate the consolidated report. This report consists of each permission name and the number of users who have either accepted or denied the demand permission (Fig. 2).

This number of acceptances and rejections are very useful for a any new user to decide whether to allow or deny the permission for the application. If more number of users have denied the permission, then the new user will know that it may not be safe to allow the permission.

Conclusion

A wide range of applications incorporate permission excursive to the application's utility. These permissions allow access to assets which are delicate in nature. This may result in the spillover of user data or utilized by the third party identified by the application. The user is unaware of this and agrees to the permissions because the user is not warned about these threats in any possible ways. So the proposed system permits the user to see that in reality how many people have genuinely agreed to this permission and how many have not. This increases the security of information and permits the user to choose which information he/she needs to share. In future, I expect up-gradation of security by evacuating those permissions which are dismissed by the greatest number of users. The application will make a request to the developer to avoid those permissions which most users have rejected in order to continue in the application market. This initiative would guarantee general safety and security of user information and counteractive action of third party applications utilizing private information.

Acknowledgment

We thank our colleagues from Christ University, Bangalore who provided insight and expertise that greatly assisted the research.

We would also like to show our gratitude to Joy Paulose, H.O.D of Department of Computer Science, Christ University, for sharing their pearls of wisdom with us during this research and we thank 3 "anonymous" reviewers for their so-called insights.

Authors Contributions

Ankur Rameshbhai Khunt: Android extra module for permission system.
P. Prabu: Allow and deny services.

Ethics

All information provided in this paper is confidential and unique. This paper has neither been published nor is under review elsewhere. In this article, we propose a new method and algorithm for the Android security. Any implementation or adaptation of this idea is subjected to the user's own result and the idea and result in this paper no way guarantees safety, security or some pre defied result.

References

Andow, B. and H. Wang, 2015. A distributed android security framework. Proceedings of the IEEE International Conference on Smart City, Dec. 19-21, IEEE Xplore Press, pp: 1045-52.
DOI: 10.1109/SmartCity.2015.207

Enck, W., D. Octeau, P. McDaniel and S. Chaudhuri, 2012. A study of android application security. Systems and Internet Infrastructure Security Laboratory.

Enck, W., M. Ongtang and P. McDaniel, 2009. Understanding android security. IEEE Security Privacy Magazine, 7: 50-57.
DOI: 10.1109/MSP.2009.26

Faruki, P., A. Bharmal, V. Laxmi, V. Ganmoor and M. Gaur *et al*., 2015. Android security: A survey of issues, malware penetration and defenses. IEEE Commun. Surveys Tutorials, 17: 998-1022.
DOI: 10.1109/COMST.2014.2386139

Fragkaki, E., L. Bauer, L. Jia and D. Swasey, 2012. Modeling and enhancing android's permission system. CyLab at Carnegie Mellon University.

Heuser, S., A. Nadkarni, W. Enck and Ahmad-Reza Sadeghi, 2014. ASM: A programmable interface for extending android security. Proceedings of the 23rd USENIX Conference on Security Symposium, Aug. 20-22, USENIX Association, San Diego, CA, pp: 1005-19.

Jain, A. and Prachi, 2016. Android security: Permission based attacks. Proceedings of the 3rd International Conference on Computing for Sustainable Global Development, Mar. 16-18, IEEE Xplore Press, New Delhi, India, pp: 2754-59.

Lee, C., J. Kim, S. Cho, J. Choi and Y. Park, 2013. Unified security enhancement framework for the Android operating system. J. Supercomput., 67: 738-756. DOI: 10.1007/s11227-013-0991-y

Luyi, X., P. Xiaorui, W. Rui, K. Yuan and W. XiaoFeng, 2014. Upgrading your android, elevating my malware: Privilege escalation through mobile OS updating. Proceedings of the IEEE Symposium on Security and Privacy, May 18-21, IEEE Xplore Press, San Jose, CA, USA, pp: 393-408.
DOI: 10.1109/SP.2014.32