Survey on Privacy Permission Management Approaches for Android OS Applications

Supriya Shinde
M.E. Student Department Computer Engineering
PCCOE, Pune University,
Pune-44, India

S. S. Sambare
Associate Prof., Department of Computer Engineering, PCCOE, Pune University,
Pune-44, India

ABSTRACT
Smart phones have been becoming popular and mobile users are increasingly relying on them to store and handle personal information. The intake of smart phone devices for email writing, online bank transactions growing with growth of the smartphone market and accessing other forms of sensitive content has lead to the emergence of a new and ever changing threat. However, recent studies also reveal the disturbing fact that users personal information is put at risk by smart phone applications. Existing mobile privacy approaches are having limitations in their performance in handling these privacy-violating smart phone applications. Android OS(Operating System) supports the third party software development mostly. However, many of these applications accesses private or sensitive data totally negligent which immediately leads to serious privacy concerns. To reduce this problem, the current Android permission rules are much too coarse and incomprehensible from the average users perspective. Mobile users does not understands the meaning of the permissions, so he or she must either accept all of them or waive the application.

Privacy Permission Management for Android OS has becomes very important security issue as, the user has only two options when they install new applications – accept the permissions and allow the install, or reject all the terms and exit without installation. So here is the need to design system in the manner which provides security to user's personal information leak and misuse of it should be prevented. This paper provides the study of different approaches of privacy permission management and the outline of all the studied papers.

Keywords
Android OS, Privacy Permission Management, security issue etc.

1. INTRODUCTION
The mobile market has expanded rapidly. Amongst mobile phones, smartphones have received extraordinary adoption. The reason behind increased demand for smartphones is the huge availability of applications that can be downloaded and installed easily on smartphones. For example, Google Play store of Google contains many Android applications to download. Along with the vendor-provided programs, third-party apps are also present on smartphone market places. In May 2013, Google announced that 900 million Android devices had been activated.[4] The increase in the levels of convenience and features of smart phones, causes a significant growth in the number of Smartphone users. For example, smart phone consists of call log consist information about the dialed and received calls, an address book with the users contact, browsing history, images, personal documents etc. As these are all private information, it has to be ensured that they don't fall in the wrong hands. According to F-Secure, a cyber-security related company, the number of new mobile threat families and variants continued to rise by 49% from the previous quarter; 91.3% of these threats targeted at Android devices in the first quarter of 2013 (F-Secure, 2013)[4].To protect personal information of user from apps of smartphones which are malicious, a new mode of privacy is needed in smartphones.[4]

This new mode can modify an apps access to user's personal information. It will be possible for user to control the access of personal information to the app in a way that, which information can be accessed and which can not. Further, the user should have run-time control to modify the previously given permission.

1.1 Background
This section specifically focuses on Android Security issues and background permission management details. Android is worked as a software stack for mobile devices .It consists of an operating system, an application framework, and core applications. Each Android application executes in a separate Dalvik virtual machine instance running as a unique user identity assigned at install time. Thus applications are essentially isolated. This design provides promising security for file accesses and limits potential damage due to programming flaws such as buffer overflow. Android restricts accesses to critical resources using permissions. A permission is simply a unique text string which can be defined by Android or third party developers. According to the documentation for Android developers, there are currently 130 permissions, which are defined in Android operating system, ranging from access to camera (CAMERA), full access to the Internet (INTERNET), dialing a phone number (CALL_PHONE), and even disabling the phone function permanently (BRICK). According to the study of Wei et al. (2012)[10], the number of Android defined permissions keeps increasing since the first widely-used release (API level3). The expansion of the permission set aims at providing finer-grained permissions controlling and accesses to new hardware features. In addition to Android defined permissions, application developers can also declare customized permissions so as to protect their own critical resources. Permissions may be required when an application is interacting with system resources, including calling system API functions, and reading from and writing to file systems.[4]

Granted permissions are assigned to an application's sandbox and inherited by all of the application's components, while required permissions are assigned to application components. In the manifest file of an application, which is included in the application package, the application declares the permissions which it requires to achieve its functionality, as well as defines the permissions for protecting its own components and resources. Figure 1 shows sample code of manifest file which shows use of <uses-permission> tag.

```
<manifest package="com.example.myapp">
        <application />
            ...
        <uses-permission android:name=
      "android.permission.SEND_SMS" />
        <uses-permission android:name=
    "android.permission.RECEIVE_SMS" />
        <uses-permission android:name=
  "android.permission.READ_CONTACTS" />
            …
</manifest>
```

Figure 1:Permission Declaration in Manifest File of Android application

A permission can be associated with one of the following four protection levels[7]:

- **Normal:** A permission risk is low which allows applications to access API calls (E.g., SET_WALLPAPER) or changing of wallpaper causing no harm to users.
- **Dangerous:** A high-risk permission which allows applications to access potential harmful API calls (e.g., READ_-CONTACTS) such leaking private users contact list or resistor over smartphone device.
- **Signature:** A permission which is granted if its requesting application is signed with the same certificate as the application which defines the permission is signed.
- **Signature-or-system:** A permission which is granted only if its requesting application is in the same Android system image or is signed with the same certificate as the application which defines the permission is signed.[7]

At install-time, a user is shown with a list of permissions which an application requests. The user must either grant or deny all of these permissions together. After the user approves the permission request and installs the application, the application owns its permissions throughout its lifetime and it does not need to request them again at run-time. Android controls Inter-Component Communication (ICC) through a reference monitor. The reference monitor provides a Mandatory Access Control (MAC) enforcement on how applications access components by evaluating whether the applications are granted with necessary permissions.

2. LITERATURE SURVEY

This section gives the brief idea about the related work, Few extensions have been proposed in the privacy security. Android Permission Extension (APEX) is a framework that allows users to specify runtime constraints to restrict application's access to resources. Users can specify constraints through a simple interface of the extended Android installer. The extensions are incorporated in Android with little changes of the code and the user interface [2].

Applications and Components: The set of applications and components in Android are denoted by A and C respectively and relative function of component and Application $ƒ: C \rightarrow A$ associates each component with a unique application. Inter-Component Communication (ICC) in Android is accomplished through the concept of Intents.

Intent: An intent is a 4-tuple$(a, б, r, є)$, where ^{a} is an action string describing the action to be performed, ^{б} is a string representing the data, r is the string representing the category and ^{є} : name$^{\rightarrow}$ val is a function that maps names of extra information to their values. The set of intents is denoted as I.

Intent Filter: An intent filter is an application's readiness to serve an intent. Intent filters are associated with individual components of applications. An intent filter association function Af : C$^{\rightarrow}$ 2f maps each component of an application to a set of intent filters where F is the set of intent filters and I $^{\subseteq}$ F. If a component c $^{\epsilon}$ C has an intent filter f, we write…

$$f \epsilon Af(c)$$

Permissions: A permission declares the requirements posed by a component for accessing it. A permission association function Ap : C$^{\rightarrow}$P associates each component to a single1 permission where P is the set of permissions. If a component c $^{\epsilon}$ C requires that a calling component have a permission p $^{\epsilon}$ P, then we write p = Ap(c).

Static Permission Function: The permission function p defines the complete set of conditions under which a component co1 is allowed to call another component co2. co1 can communicate with co2 if and only if either 1) there is no permission associated with the second component or 2) the application to which co1 belongs has been granted the permission required by co2. Mathematically:

$p(co1, co2, i) \Leftrightarrow Ap(co2) = null \vee$

$\exists p \epsilon P, a \epsilon A \ .a = ƒ(co1)$

$\wedge p = Ap(co2) \wedge p \epsilon \mu(a1)$

$\wedge i \epsilon Af(co2)$

Dynamic Permission Function: The dynamic permission function specifies the conditions under which a component co1 is granted permission to call another component co2 using intent i. It incorporates the static checks as well as the dynamic runtime constraints in its evaluation. To grant the permission, Android's permission checks must grant the permission and there must not be a policy that denies the permission. [2]

Mathematical Form:

$p(co1, co2, i) \Leftrightarrow Ap(co2) = null \vee$

$\exists p \epsilon P, a \epsilon A \ .a = Ç(co1)$

$\wedge p = Ap(co2) \wedge p \epsilon \mu(a1)$

$\wedge i \epsilon Af(co2)$

$\wedge \Leftrightarrow \neg \exists l \epsilon \Lambda \ . l(a. p) = deny$

Mock-Droid is also a modified version of Android. It allows users to 'mock' application's access to resources. Mocking means that resources are reported as empty or unavailable whenever an application requests access. In contrast to Android's static permission system, users can revoke access to specific resources at run-time [3].

TISSA stands for Taming Information Stealing Smart-phone Applications. TISSA is a system that implements a privacy mode that permits users to flexibly control application's access to personal information in a fine-grained manner. Access grant can be dynamically adjusted at runtime as per users permission setting of application [4]

LBE Privacy Guard provides a back-ground service that constantly monitors applications' activities. Users get notified whenever an application attempts to access a sensitive re-source

like the location, the phone ID or the Internet. The requested access can then be permitted or denied by the user using LBE Privacy Guard Application.[11]

A central aspect of the PMP (Privacy Management Platform) is to enable applications offering various Service Features that build upon user acceptable permission sets. To achieve this, an application has to specify various Service Features with mandatory private data requirements. PMP informs the user about the available Privacy Settings and their consequences for the scope of services. The most outstanding features of PMP are its extendibility in terms of adding user generated new Resources in arbitrary granularity - its context sensitivity as permissions can be granted or revoked according to the current situation and its user interaction as a user is always informed about the impact of any decision both on privacy as well as on application's service. [6]

3. COMPARATIVE ANALYSIS

This section describes the comparative analysis on related approaches based on following parameters and advantages, disadvantages of each approach in detail.

1) Blocking of data: Permission access to specific resource may blocked through security exception. It depends on application how it responds to such situation. It may crash, exit or continue to run.

2) Mock display of data: It is an alternative to blocking of data. Access to a resource is to provide mock or dummy data value.

3) Logging Access: Logging or Log details are useful to find out which resources are actually used by which application.

4) Policy Control: Control of resource access is based on policies by the system. The end-user does not have control over permissions other than to change policies of the system. This is in distinction to manual control, where the user has direct control over the permissions of single applications.

5) Configuration Time: It will give information about the type of change and save permission settings supports by each of these approaches. If it is static means application supports installation time permission settings. Another can be at Runtime changes of permission changes supported by that particular application.

Table 1. Comparative analysis

Approaches / **Parameter**	**Apex**	**Mockdroid**	**TISSA**	**LBE Privacy Guard**	**PMP**
Blocking of data	Blocks data if required	No blocking of data	No blocking of data	Blocks data info	Blocks data as per required
Mock data display	No mocking of data	Fake or mock data value display	Bogus or Anonymized data value display	No	No dummy data display
Logging Access	No	No	No	Display and use of log resources	No
Policy Control	No	No	No	No	Yes
Modified Android Architecture	Yes	Yes	Yes	NA	Extension to existing architecture.
Configuration Time	Runtime	Installation/ Runtime	Installation/ Runtime	NA	Installation/ Runtime
Crash Safe	No	No	Yes	Yes	Mostly yes

From above Table 1, It has been observed that PMP approach gives better Functionality among the other described approaches in terms of Runtime change settings facility, Policy control to the user, Blocking of data as per user's requirement.

Table 2: Comparative Study Of Approaches

Approach	**Advantages & Disadvantages**
Apex	**Advantages** 1) The user even can install an application, although if only a subset of the requested permissions is granted. 2) However, at runtime permissions can be added, modified, or removed. Certainly the biggest improvement is undeniably that the user can define various constraints under which certain functions may not be executed. **Disadvantages** 1) Applications might crash if they try to access data though the user has withdrawn the therefore required permissions.
TISSA	**Advantages** 1) This privacy mode provides the facility to grant access dynamically by adjusting permissions at runtime according to the users need. **Disadvantages** 1)It works only for location, phone identity and does not work for SMS ,Image Gallery, Bluetooth related permission access etc.
Mockdroid	**Advantages** 1) It allows the user to control the collection and distribution of personal data, which is a great contribution to privacy on mobile devices. **Disadvantages** 1) The user is able to use third party apps, though with the lack of some functionality that is dependent on mocked data. 2) As the title implies, providing mocked data means that the applications may lose important functionality.
PMP Model	**Advantages** 1) It is extendable, context sensitive and fail safe mechanism to enable the user to become master of his or her private data. **Disadvantages** 1) Application developer has overhead of defining service features. 2) There are still some missing features are present which need to be handle in future.

3.1 Proposed Model

In proposed model modifications over existing Android Architecture will be implemented which will provide user flexibility to control their resource accessibility. It will provide alert messages whenever any installed applications try to access your private data such as phone call log information, internet history, gallery etc. It will provide facility to change and save permission settings as per users choice .It will provide better security mechanism to all android smart phone users.

4. CONCLUSION

This paper gives detailed survey and analysis on existing approaches in Android market. Among the existing approaches explained in this paper PMP approach gives better performance from comparative analysis. The proposed solution design will provide ability to fine tune their privacy preferences by turning on and off permission usage of individual application which provides security to user's sensitive data. Future work includes resolving the issues of less user friendliness ,No blocking of data and user privacy control through extending proposed model settings design, alert messages display functionality, user friendly access on overall usage.

5. REFERENCES

[1] Alastair Beresford, Andrew Rice, Nicholas Skehin and Ripduman Sohan. MockDroid: Trading privacy for application functionality on smartphones. Computer Laboratory, University of Cambridge, 2011, www.cl.cam.ac.uk/~acr31/pubs/beresford-mockdroid.pdf

[2] Vincent Freeh, Xuxian Jiang, Xinwen Zhang and Yajin Zhou. Taming Information-Stealing Smartphone Applications (on Android). Department of Computer Science, NC State University, 2011, www.csc.ncsu.edu/faculty/jiang/pubs/ TRUST11.pdf

[3] Michael Kern, Johannes Sametinger,Permission Tracking in Android.UBICOMM 2012 : The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies

[4] Zheran Fang ,Weili Han .Yingjiu Li, Permission based Android security: Issues and countermeasures, computers & security :2014

[5] Mohammad Nauman and Sohail Khan. Design and Implementation of a Fine-grained Resource Usage Model for the Android Platform. Department of Computer Science, University of Peshawar, 2010

[6] Christoph Stach and Bernhard Mitschang: Privacy Management for Mobile Platforms -A Review of Concepts and Approaches .In: 2013 IEEE 14th International Conference on Mobile Data Management(2013)

[7] http://developer.android.com/guide/topics/manifest/permission element.html

[8] Ryan Farmer:" A Brief Guide to Android Security"(2012)

[9] Jialiu Lin:"Understanding And Capturing People's Mobile App Privacy"(October 28, 2013)Carnegie Mellon University.

[10] Wei X, Gomez L, Neamtiu I, Faloutsos M. Permission evolution in the android ecosystem. In: Proc. of ACSAC. ACM; 2012

[11] Google Play. LBE Privacy Guard. play.google.com/store/apps/details?id=com.lbe.security.lite.