Security on Android Apps

Ritu Sheoran¹, Meenakshi Chaudhary²

¹M. Tech Scholar, Department of Computer Science and Engineering, Seth Jai ParkashMukand Lal Institute of Technology, Radaur (Yamunanagar)
²Assistant Professor, Department of Computer Science and Engineering, Seth Jai ParkashMukand Lal Institute of Technology, Radaur (Yamunanagar)

Abstract — For the smartphone users the biggest problem is privacy and the reason behind this is sharing of sensitive information while accessing the numerous apps. However, it is very difficult to understand which application can access what kind of information and to achieve the functionality what are the minimum number of accessing approvals are required. Before the installation of an application, all the permissions which are requested by the apps are represented in the dialog box. And it depends on the user what to do in such condition either to accept the permissions requested or select not to continue with installation. Even the security risks are also associated with online advertisements. It is possible that to disperse malicious malware with online promotions and taint end clients' PC. The individual information is gathered by all internet publicizing organizations without clients' knowledge. Individual data can be utilized to damage clients or can be sold to outsiders. In this paper we build up a wrapper that give a consent administration interface furthermore build up a Patcher that is being infused into the APK installer record to improve the security of the Android applications.

Keywords — Android, Malware, Manifest files, Permissions, Android Security, Advertising banners.

1. INTRODUCTION

1.1 OPEN SOURCE

Android was introduced in 2007, alongside the establishing of open handset organization together that is comprised of software, hardware and telecom organizations invent open standards for cell phones. There are millions of android app developed and downloaded from Google play store. For cell phones android is open source software. In the development of life cycle of an application android never force strong controls as per the philosophy of design, namely openness. The source codes are discharged to general society. Anyone can compose and put applications without affirm by Google. However the absence of control can bring about vulnerabilities particularly when the security of the framework relies on upon the client validation choice. Android permit everybody to compose their own particular application and afterward distribute it unreservedly.

1.2 FEATURES OF ANDROID

- Interface
- Applications
- Memory management

1.2.1 INTERFACE

Android's default client interface is normally centered on direct control, utilizing touch movements that frequently compare to genuine activities like pinching to control the screen stuffs, tapping, alongside virtual keyboard. Game controllers and in addition physical consoles are supported by means of USB or Bluetooth. At the point when the client inputs then the reaction is intended to be prompt and gives a liquid touch interface, over and over utilizing the vibration capacities of the gadget to give criticism to the client. Interior equipment, for example, accelerometers, gyroscopes and proximity sensors are utilized by different applications to react to the client activities, for instance permitting the client to drive the vehicle in racing games by pivoting the gadget, control of a controlling wheel.

1.2.2 APPLICATIONS

To increase the functionality of a gadget the application used are generally written by using Android software development kit (SDK) and, mostly, the android APIs are completely accessed by java programming. There is a large collection of third party app’s in android, and users can get these...
applications by just downloading the android application package (APK) file and then installing them or one other way is utilizing an application store by which user can download, install, remove and update the applications from their gadgets. The primary store which is installed in android gadgets is Google play store. Android users can download, install and remove any of the app present in Google play store either it is made by Google or third party inventor. In play store there are more than one million apps for android clients.

1.2.3 MEMORY MANAGEMENT
Powersource for the android gadget’s is battery, the designing of android gadgets are done in a way that the power consumption of the processes remains at minimum. When the application is ideal means that it is not in use then the system suspends its operation so that, while available for instant use rather than close, it does not use CPU resources and battery power.

Android manages all the applications that are stored in the memory automatically: when the memory is low than the system will begin unnoticeably and automatically closing all the inactive processes, starting with those that are inactive for a long time.

1.3 Android application framework
1.3.1 Application Components:
Components are fundamental building blocks of Android applications. Each component can be run individually, either by system upon permitted requests from other applications or by its embodying application. Android applications can comprise four types of components [1]:
- Activity Components
- Service components
- Broadcast Receiver
- Content Provider

(1) **Activity components**: offers the origin of the Android user interface. Each Application may have numerous Activities showing dissimilar screens of the application to the user.

(2) **Service components**: these components provide background processing capabilities, and do not offer any user interface. Playing a music and downloading a file even as a user interacts with another application are the examples of operations that may run as a Service.

(3) **Broadcast Receiver components**: react serially to system-wide message broadcasts. A receiver component usually acts as a gateway to further components, and passes on messages to Services or Activities to handle them.

(4) **Content Provider components**: make available database capabilities to additional components. Such databases can be mostly used for both intra-app data persistence along with sharing data across applications.

1.4 Permissions
Enforcing permissions is the other mechanism, other than sandboxing, provided by the Android framework to save applications, by which limitations are placed on the specific operations that an application can perform, such as interacting with the databases and the system APIs, along with cross application interactions. Each application should state upfront as part of its manifest the permissions it needed, and the Android system prompts the user for consent during the application installation. Should the user decline permit the requested permissions to an application, the application installation is dropped. No dynamic mechanism is given by Android for granting permissions after application installation. The manifest file also declares permissions imposed by the application or by any of its components, the other applications thus should have those permissions to interact with such protected components [1].

1.4.1 Permission Protection Level
Android permissions manage the access to functionalities and sensitive resources. There are nearly 134 Android-defined permissions which are accessible to third party applications. Permissions are mainly defined with one of the four dissimilar protection levels, which differentiate the potential risks implied in the permission and enforce conflicting install-time approval processes. Protection levels which are associated with permissions are [11]:
- Normal
- Dangerous
- Signature
- SignatureOrSystem
Normal: normal permissions are permitted automatically.

Dangerous: permissions can be granted by the user during installation. If the permission request is rejected, then the application is not installed.

Signature: signature permissions are only permitted if the requesting application is signed by the same developer that defined the permission. Signature permissions are helpful for controlling component access to a small set of applications controlled and trusted by the developer.

SignatureOrSystem: These permissions are granted if the application is installed in the system applications folder or if the application meets the Signature requirement. System applications folder cannot place the applications from the Android Market. System applications must be pre-installed by the device manufacturer or manually installed by an advanced user.

Aside from the above mentioned division, permissions are also divided by the way of accepting them:

- **Time-of-use**: user must confirm this permission when executing sensitive operation (e.g. access to device location). It is the only way to prevent applications to access device resources.

- **Install-time**: accepted when installing application, user accepts them as one; he cannot choose which permissions to accept and which to deny [14].

### 1.5 Android Security model

The android security model is planned in such a way that no application is given the permission to carry out any operation that can poorly affect further applications, the user or the operating system. Android is considered to be a multi-process system because each application runs in its own process. Linux facilities such as group and user IDs are allocated to applications which enforce security between the system and the applications at the process level. Additional security is provided using the “Permissions” mechanism that grant each process to perform certain operations only and also provides “per-URI permission” that give improvised access rights to certain information only.

Permission mechanism and Sandbox is said to be the base of the Android Security Model. It is known that each application’s code runs in a specific Dalvik Virtual Machine which has a specific user ID assigned to it. Each application runs in separation from the other applications and hence any application has no access to another application’s files [17].
2 Related work

2.1 Xiaolei Li, Guangdong Bai et. al (2014)

Mobile devices are becoming more and more general purpose, and therefore the physical boundary used to divide important resources disappears. Accordingly, malicious applications get chances to misuse resources that are presented on the mobile platform. In this paper, the author proposes resource virtualization as a security mechanism for the Android system to make stronger the physical barrier between several types of resources and enclose resource-abusing Android apps. The physical resources on a mobile device are virtualized to a dissimilar virtual view for chosen Android apps. Resource virtualization simulates a partial but stable virtual view of the Android resources. Therefore, it can never enclose the resource-abusing apps efficiently, but also ensure the usability of these apps. Implement a system prototype, RVL, and evaluate it with real-world apps of different types. Their results demonstrate its effectiveness on malicious Android apps and its usability and compatibility on benign Android apps [5].

2.2 HUAN ZENG, YAN REN et. al (2014)

In this paper, through studying the Android permissions system and Android API system, relationship is found out between permissions and APIs. Furthermore, an approach is proposed to detect message intercepting malware. The involvement of this paper is threefold: first, perform static analysis on the app to take permissions and system APIs. In order to avoid APIs and permissions over declaration, java function call graph is build and find the system APIs been used; secondly, a light weight method is purposed to dynamic find out and update the relationship between permissions and APIs; third, a dynamic controlled method is purposed to detect android malwares. This proposed method is verified by extensive experiments [6].

2.3 Andre Egners, Bjorn Marschollek et. al (2012)

Permission models have become ordinary on smartphone operating systems to manage the rights permitted to installed applications from third party. Earlier to installing an app, the user is typically presented with a dialog box present the permissions which are requested by the app. The user has to choose either to accept all of the requested permissions, or opt not to continue with the installation. The majority of ordinary users are not able to fully understand which set of permissions allowed to the application which may be harmful. In addition to the knowledge gap between application programmer and user, the missing alterability and granularity of most implementations of permission model help an attacker to circumvent the permission model. In this paper Author focus on the permission model of Google’s Android platform. Permission model is detailed, and show a selection of attacks that can be composed to fully cooperation a user’s device using inconspicuously viewing applications requesting non-suspicious permissions [14].

2.4 Yajin Zhou and Xuxian Jiang (2012)

The popularity and acceptance of smartphones has significantly stimulated the increase of mobile malware, especially on the popular platforms like Android. In light of their rapid growth, there is a need to develop efficient solutions. However, defense capability is mainly forced by the limited understanding of these emerging mobile malware and the shortage of timely access to interrelated samples. In this paper, Author focus on the Android platform and aim to characterize or systematize existing Android malware. Mainly, with more than one year effort, there is a collection of more than 1,200 malware samples that wrap the majority of existing Android malware families, ranging from their entrance in August 2010 to recent ones in October 2011. In addition, systematically distinguish them from a variety of aspects, including their activation mechanisms, installation methods along with the nature of carried malicious payloads. The evolution and characterization-based study of representative families reveal that evolving rapidly to circumvent the finding from existing mobile anti-virus software. Based on the evaluation with four
representative mobile security software, experiments show that the best case detects 79.6% of them while the worst case detects only 20.2% in the dataset. These results evidently call for the need to better develop next-generation anti-mobile-malware solutions [15].

2.5 Wook Shin, Sanghoon Kwak et. al (2010)
This paper shows a flaw in the permission scheme of Android. The Android framework imposes a permission-based security policy where an application can retrieve the other parts of the system only when the application is exactly permitted. The security of the framework mainly depends on the owner of a device since the authorization decisions are mostly made by the user. As a result, the permission scheme forces much of the administrative burden on the user alternatively of keeping it simple. Moreover, the framework does not apply enough controls nor maintain dynamic adjustment in the following respects, No constraint or naming rule is applied for a new permission declaration. Once an application obtaining a permission, the permission is never recall during the lifetime of the application. Two dissimilar permissions can be in use having the same name. These characteristics of the framework can result in a security flaw [16].

3 Proposed work
When it comes to restricting app permissions android takes an "all or nothing" approach. We have no granular control over which data apps can access. So the only way is to not install the app in the first place to prevent an app from seeing your location, accessing contacts, revoking face book login or Google plus login. Using modes we can revoke individual permissions for installed apps, but these all had one thing in common they required root access. Granular permission control system that does not require root.

- Existing apps can take any of our apps, than inject a wrapper into their coding that allows us to disable particular permissions. It's a rather ingenious solution, since its only requirement is that we've enabled Unknown sources in our phone's Security settings.
- A more technical standpoint, a wrapper that will give a permission management interface is being injected into the APK installer file for the selected apps. We can revoke the app's permissions to run on startup, Select a permission, than choose either "Prompt" or "Deny." These apps will not be able to access the permissions that we have revoked. Uninstall the original copy of this app, than install the modified version immediately after.

4 RESULT
While installing any application there requested permissions are displayed in the appearing dialog box. Android takes "all or nothing" approach. User choice is to accept all the permissions to install the app or deny to cancel the installation. To prevent an app from being seeing our personal information like location, accessing contacts we disable certain permissions to enhance the security of the app.

- In the above permissions we disable certain permissions like Bluetooth, Telephony etc. So that we can enhance the security and protect our personal information from being send to 3rd party apps.
- Security risks are also associated with Smartphone advertisements. There are various advertising ads which are displayed in our mobiles along with apps. The banner ads are of various formats. If there is any click on these ads then it will take on another webpage. It is probable to distribute malicious malware with online ads and infect end users’ computer. We can remove these advertising adds by using Patcher.
5 CONCLUSION
In this paper we design Wrapper and Patcher. The purpose wrapper is to disable certain permissions before the installation of the requested app. A wrapper will provide a permission management interface which is injected into the APK installer file for the selected apps. We can revoke the app's permissions to run on startup. Select permissions, than choose either "Prompt” or "Deny." These apps will no longer be able to access the permissions that we have revoked. The original copy of this app is uninstalled, after this the modified version will installed immediately. Even we can set the permissions of an app before they are downloaded. This will results in not completely disabling the app, but will allow users safeguard their privacy and keeps apps from accessing any more user data. For example a gaming app from 3rd-party store need access to your Google account, calendar or modify your contacts list. The purpose of Patcher is to provide a little private area to apps taken from untrusted sources. Patcher will not allow the app to play with rest of the device that stores our confidential information. The main purpose of Patcher is to resize the Advertising Banners to zero and this is efficiently implemented. Results are evaluated and resulting apps are highly secure because advertising banners are zero. This provides more protection against privacy violation of user’s personal data.

6 REFERENCES
[1] Hamid Bagheri, Alireza Sadeghi, Joshua Garcia and Sam Malek, Member, IEEE, “COVERT: Compositional Analysis of Android Inter-App Permission Leaking” 0098-5589 (c) 2015 IEEE.


[3] Xing Gao, Dachuan Liu, Haining Wang, Kun Sun, “PmDroid: Permission Supervision for Android Advertising” 1060-9857/15 $31.00 © 2015 IEEE.


[8] Wei Wang, Xing Wang, Dawei Feng, Jiqiang Liu, Zhen Han, and Xiaoliang Zhang, Member, IEEE, “Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection” 1556-6013 © 2014 IEEE.


[13] Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Peris-Lopez and Arturo Ribagorda, “Evolution, Detection and Analysis of Malware for Smart Devices” 0000–0000/00/0000.00 © 2013 IEEE.

[14] Danyang Jiang, Xiangling Fu, Maoqiang Song, Yidong Cui, “A SECURITY ASSESSMENT METHOD FOR ANDROID APPLICATIONS BASED ON PERMISSION MODEL” 978-1-4673-1857-0/12/$31.00 ©2012 IEEE.


