ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(14), DOI: 10.17485/ijst/2016/v9i14/87071, April 2016* ISSN (Online) : 0974-5645

Security Comparison of Android and IOS and Implementationof User Approved Security (UAS) for Android

Dar Muneer Ahmad[1*] and Parvez Javed[2]

[1]National Institute of Electronics and Information Technology (NIELIT), SIDCO Electronics Complex, Old Airport Road, Rangreth, Srinagar - 191132, Jammu and Kashmir, India; muneer@nielit.gov.in

[2]Department of Computer Science, University of Kashmir, Hazratbal, Srinagar - 190006, Jammu and Kashmir, India; javed_parvez@kashmiruniversity.ac.in

Abstract

Background/Objectives: The widely spread Smartphone operating systems i.e. iOS and Android are loaded with the inherent security measures to protect their millions of users worldwide. Android, being familiar and popular in open supply mobile package i.e. open source platform has some security limitations or vulnerabilities which are also present in iOS that is owned by Apple and is a proprietary platform with some open supply parts. **Methods/Statistical Analysis:** We compare the basic protection options of Android and iOS, with the objective to combine the user approved security (UAS) model in Android so that we are able to grant permissions to access various resources on a Smartphone at the time of execution. **Findings:** In this paper we propose the implementation of a reverse engineering method that restricts Access of the resources and provides a requirement based procedure to access resources. **Applications/Improvements:** By incorporating the user approved security, the repackaged app would be more secure and will be capable of running on all the devices.

Keywords: Android, App, iOS, Mobile Package, Open-Supply, Repackaged App, Reverse-Engineering

1. Introduction

The leading Smartphone operating systems i.e. Android and iOS, are claiming to have more than 50 thousand applications (apps), are in a run to dominate each other in terms of user friendliness, accessibility and on other fronts.With a vast variety of new and innovative Android and iOS apps constantly coming into their respective markets, the common user is unaware of the risks these apps bring in with them. User privacy is of paramount importance to the Researchers and they are in continuous search to find out the better Smartphone operating system which is capable to handle the privacy of the users in terms of securing users' confidential data. Researchers found that compared to Android, iOS is a safer platform. It has been found that 79% of malware dominantly attacked Android and only small percentage of 0.7% of all mobile malware targeted iOS. Researchers[1-4]securing iOS and Android found that the existing security frameworks need enhancements in order to safeguard their novice users. OS improvements have been done by incorporating the third party frameworks[5] which resulted in the enhanced user-friendliness of the existing security permission system but have several usability issues as a result of the modification of the existing security framework[6,7].

A novel technique of improving or enhancing privacy is to disable the apps permissions completely, thereby restricting the app's accessibility to the resource of a Smartphone and providing the users with more control over the resources, at the time of execution, by selectively granting or restricting the access by the user. The technique reverse engineers the app and grants the

* *Author for correspondence*

permissions at run time. Researchers researched on the removal of permissions and controlling the permissions to increase the Smartphone security and privacy, both of the enhancements will be taken care of by the authors in this paper.

The next section compares the world's dominant Smartphone operating systems i.e iOS and Android with the intension to find out the limitations of the security features of these two platforms. In section 3, we try to find out the work done by the earlier researchers in securing the novice users. In section 4 we incorporated our novel technique and provide a overall architecture of user approved security system. In section 5 we present the implementation of the proposed User Approved Security system. Finally, we draw our conclusions and identify the future enhancements in the last section.

2. Security Comparison of Android & IOS

Smart phones with Android operating systems have their own security framework and areLinux-based.If we compare the desktop operating system where the same UID is used to run various applications, apps run on Android are individually separated from oneanother and are run under distinct UIDs with individual processes. Each of the process is given distinct set of permissions to provide access to different resources of the smart phone. It is left to the programmer to give the required permissions in the android app. Under different UIDs the apps can't read/write other apps data or files and it has to be done by the programmer programmatically[8]

The core mechanism of security in Android is based on permissions, called as permission based security for restricting the access of various resources or files. The programmer is given the liberty to allocate these permissions to various resources and is left to their own understanding of the app. This openness of giving more freedom to the programmers is at the cost of more malicious attacks and makes the framework more vulnerable to the malware. In Android the app can be accessed/downloaded from the Google market or any other play stores on the web, during the installation the user is asked to accept the list of mandatory permissions in order to successfully install the app. User is left with no choice to selectively accept or reject any of the permission. In order to successfully install the app all the permissions are to be accepted.

Android Security framework prompts the user to accept the list of mandatory permissions, during the installation process and is defined in a separate file called Android Menifest.xml. The major drawbacks of this framework are as under[9]

- The permissions are to be accepted as the user can't pick one and leave other while installing the application. A malicious app may ask for out of context permission to access critical data and user is left with no choice other than to follow all the permissions in order to successfully install the application.
- For a common user it is difficult to understand the appropriateness of the permissions for the app they want to install. In some cases a novice user can understand whether the permission is legitimate or not, for example when a CHESS gaming app requests the privilege to access the SD card in a Smart phone or to access Contacts.
- As the permissions are left to the programmers own understanding, the functionality of the app can be achieved with less number of permissions or none at all.

The smart phones with iOS are not dependent on the users for their permissions and all the permissions are implicitly granted. The apps are fully capable of accessing any of the resources within the smartphone and the user is unaware of the background activities that may be taking place.Chin et al.[10] selected thirty Android users and thirty iOS users for testing, It was found that Android users had more free of cost apps than iOS users downloaded from different markets. It was also found that, around twenty percent of Android users seriously consider the list of permissions while installing the apps, and additional forty percent stated that they sometimes carefully check the permissions while installing the app. On the other hand Felt et al. found that only seventeen percent of Android users pay serious attention to the list of permissions during the installation of the app. While comparing thirteen Android and eleven iOS users to find their expectations and concerns related to privacy King[11] found that the iOS users are more confident because of the internal review process done by Apple. However, the researchers, King and Kelleyetal[12]. found that users of Android often trust that Google rechecks the apps with

regard to the legitimacy of permissions and the malicious behavior of the app. The users of both Android and iOS, who trust their operating system providers felt safe when downloading and using different apps as they feel that the apps are examined properly. From this point of view there is no substantial difference between the platforms. The users of Android and iOS are notified differently about the usage of the resources accessed by the apps. The users of iOS are not aware of the resources accessed by the apps and they trust the privacy policy of the operating system. On the other hand Android users can go through the list of permissions at the time of installation but some of the novice users are not able to understand the permissions they are accepting. Researchers[13] found that most of the users are not able to understand the permission screen and don't care about the permissions.

The protection level of these two leading smart phone operating system is given in the table below.

Table 1. Protection levels of android and iOS against various malicious attacks

Protection against	Google Android	Apple iOS
Malware	25%	100%
Internet based attacks	100%	100%
Engineered Attacks	0%	0%
Resource and Service based Attacks	50%	75%
Loss of Data	25%	50%
Attacks based on data integrity	25%	50%

3. Related Research Work

It has been found by the researchers, Pridgen& Wallach[14] who tried to find out the number of permissions asked by 114,000 numbers of apps. They came to the conclusion that the number of legitimate permissions in apps permission screen is less and apps have more unnecessary permissions with the result programmers are making insecure apps and privacy of users is compromised.The programmers are keeping a provision for the inclusionof advertisements so as to earn from these advertisements, is one of the major reasons for the extra number of permissions, researched by Dietz, Shekhar & Wallach[15;] the advertising requires additional resources to be used that are why the more number of permissions are required by the programmer. The novice users are innocent and they hardly care about the permissions they are asked researched by Felt et al[16].and Kelley et al[17]. As such the security and privacy provided by the permission system may not be adequate for the novice users. It was recommended by Felt et al. to enhance and modify the Android OS. He also suggested educating the novice users about the security risks if he does not go through the list of all the permissions and blindly accepts all the permissions. Kern & Sametinger[18] researched in a different way by recommending the creation of separate accessibility control on separate app i.e the user can select the legitimate permissions and reject other permissions based on his understanding out of the list of all the permissions. They also recommended the use of third party apps as extensions to the OS so as to properly take care of app's permissions, and designed their own app as a proof of concept that gives a user more freedom to allow or disallow a request as it comes. Researchers recommended the app permissions to be removed so as to enhance the security of a common user. The app was modified to give the detailed view of all the permissions so that the user is able to view the permissions and then can be removed or retained. This requires the extraction of an app's source codeor the app is de-compiled, modifications are done to remove these unwanted or extra permissions and then re-compiled with less adequate number of permissions. Researchers also found that while it is technically feasible tomanually make modifications and remove some unnecessary permissions manually from an application via the AndroidManifest. xml, it most of the times resulted in an app not working properly during the process of extraction, modification and recompilation. Berthomeet al[19]. proposed to use a combination of two apps, the Security Monitor and the Security Reporter. In this case the targeted app can be monitored and all the log details can be provided.Dawu, Juanru& Yuhao[20] researched in a same approach i.e by reverse engineering the Android applications to guide them inunderstanding the malware targeting Android. Our research in which we propose and implement the User Approved Security (UAS), which empowers the user with the additional control on the permissions based on his requirements, is an efficient and effective system as it additional control in the hands of common users without modifying the Android OS which may result in other issues[21-23].

4. Proposed UAS System

The objective of User Approved Security (UAS) is to

provide the novice users with more control over the permissions at the time of execution in order to enhance their security. The detailed architecture is given in Figure 1 below. The users are informed about the access of any of the resource within the smart phone by giving more control to the user; he/she can allow or reject the permission at run time. The User approved security system extracts various files from the APK file by using the reverse engineering. One of the important file called AndroidMenifest.xml lists all the permissions required by the app. The permissions are checked one by one and unwanted or extra permissions are removed. In order to understand the working of the system lets examine the app called 1mg, the one health app you must always have on your phone, available on Google Play. The app gives all the details related to any medicine, online consultation and we can also buy medicine using this app. The important permission this app requires is the INTERNET and we have to grant that permission. If this app asks for MANAGE_ACCOUNTS permission that the app needs to get all the details regarding the user account, our system will remove such unnecessary permissions.Once the extra permissions are removed from the app which is present in AndroidMenifest.xml file, the second step that our system takes is to prompt the user with the permission that the app is trying to access. The user may allow or disallow the access of the resource. With this system the user is given more liberty to restrict the app which may be doing any malicious activity.

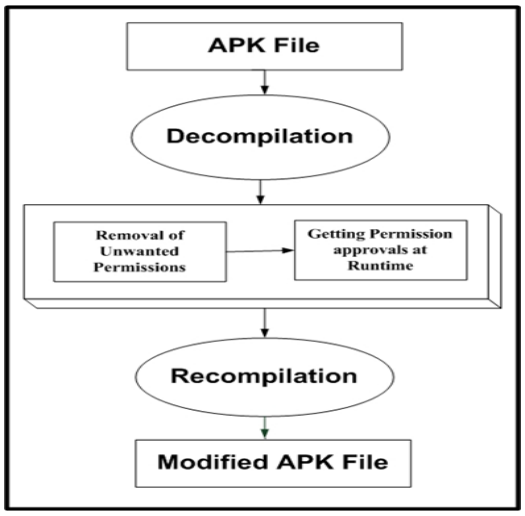

Figure 1. Architecture of UAS System.

The UAS system does not require modifying the Android operating system, which may result in usability issues. This approach of permission removal and getting permission approvals at run time is relatively new and emerging approach, as it does not modify the Android operating system.

5. UAS System Implementation

- *Reverse Engineering.*

After compilation, each Android app is packaged in a single file called Android Application Package (APK) file. This file includes all source code and resource files called the dex files. The name of this file is given by the programmer and has the .apk extension.One of the important file in the Android project structure is the AndroidMenifest.xml which contains all the necessary information about the app including the list of permissions. In order to implement the User approved security, we have to extract all the files using reverse engineering. The contents of the apk file are given in the figure 2 below.

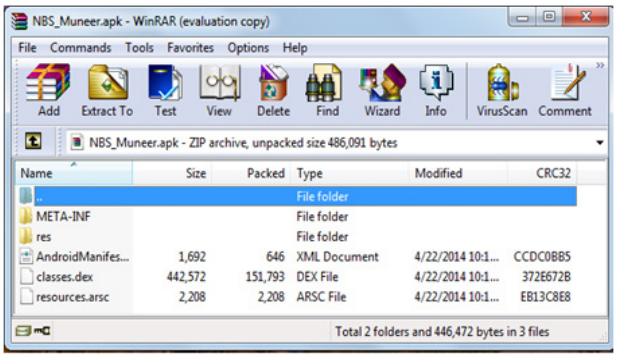

Figure 2. Contents of the APK file.

In order to extract various files from the apk file we have to download the following files from the web

- classes_dex2jar.src
- jd-gui-0.3.5.windows
- SignApk
- apktool1.4.1.tar
- apktool-install-windows-r04-brut1.tar

Extraction and Recompilation:The AndroidMenifest. xml and other source code files are extracted from the apk file from a file for example Muneer.apk using the following commands: "apktool if Muneer.apk" and then type "apktool d Muneer apkpath_output".

Once we extract the various files from the .apk file, our system will remove the unwanted and extra permissions

from AndroidMenifest.xml file. Now we have to recompile the repackaged app to create a modified .apk file with UAS system in place. We use the nextcommand i.e"apktoolb path2_recompiled.apk" and a new folder "build" is created in the decompiled path. The new apk file will reside on the same path. Still the modified app would not work and when we try to run it into system, the ROM will be hold up in BOOTLOOP XD, In order to come out of this we need to use signapk.jar and type "java –jar signapk.jar certificate.pem key pk8 our_repackaged_new_apk".

We are also in need of higher level of code, as such we type the commands "jar xvfMuneer.apkxxxx dex" and"d2j-dex2jar xxxx.dex" and we have xxxx-dex2jar file.By using dex2jar, we have to type in the following command: "jar xvf xxxx-dex2jar.jar". After the execution of the above commands, two folders will be generated; one top level package name and other one is will be the Android.

Now to get the original java code from the .class files just extracted we need the Java de-compiler.

- Unwanted Permissions Removal.

As the AndroidMenifest.xml lists all the permissions, we can remove the unwanted permissions from this file. Before removing the permissions we have to take care that this permission will not affect the overall functionality of the app. For example the app called 1mg, the one health app you must always have on your phone, available on Google Play. The app gives all the details related to any medicine, online consultation and we can also buy medicine using this app. The important permission this app requires is the INTERNET and we have to grant that permission. If this app asks for MANAGE_ACCOUNTS permission that the app needs to get all the details regarding the user account, our system will remove such unnecessary permissions. Once the extra permissions are removed from the app which is present in AndroidMenifest.xml file, the second step that our system takes is to prompt the user with the permission that the app is trying to access.

The objective, therefore, is to remove unnecessary or extra permissions from an app that should not be required. It is very important to understand the domain of the app based on which the permissions of the app can be removed if needed. For example social networking apps cannot run without INTERNET so we cannot remove that permission from such apps. We have to take care in removing the permissions as some apps may be so tightly integrated with a certain permissions that the app may become non functional without it.

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission />
```

Figure 3. AndroidManifest.xml with list of Permissions.

RuntimePermission approvals.

As we have already extracted the various files of our app by implementing the reverse engineering in the previous sections. The unwanted permissions are already removed and we are left with some legitimate permission which is required to run the application properly. Now we give additional control to the user at run time to decide whether the resource should be accessed or not. For Example if our app is trying to access the EXTERNAL_ STORAGE, trying to read the contacts, trying to send the SMS or trying to read the IMEI number, the user will be prompted with a dialog box with the information about the activity the app is trying to do. This gives the user additional power to stop or allow the access at run time. In case of our earlier example of 1mg app, the unnecessary permissions are removed in the first stage and we have some permission left which are mandatory for that app to run properly. Now the next level of protection is given the user and if the 1mg app tries to access the INTERNET, it prompts the user with allow or disallow dialog there by giving more flexibility to the user.

We have to thoroughly understand the code and try to find out the block of the code where the permission is actually granted. We have to enclose that block with the condition that if the user allows then only the code should get executed. For example if our app is trying to read or write EXTERNAL_STORAGE for which the permission is already given by the user at the time of installation we have to locate the lines of the code trying to read or write on the SD card and enclose those lines of code with the condition that if the user approves, only then the SD card should get read or written.

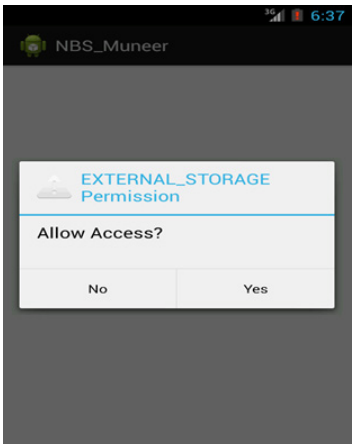

Figure 4 Run time permission approval.

6. Conclusions & Future Work

We compared the world's dominant Smart phone Operating Systems viz. Android and iOS to get the detailed security loopholes and advantages of one over other. Our research focused on the privacy issues concerning the novice users and their unawareness of the permissions they are granting at the time of installation. We attempted to remove the extra and unnecessary permissions from the app thereby giving second level of security to our users. Not only we removed the unnecessary permissions, we also gave our users the authority to approve or reject the permission at run time without modifying the Operating System.

The manual implementation of User Approved Security (UAS) has been undertaken in this research. All the steps undertaken in this research i.e de-compilation, permission removal and re-compilation were done manually. Therefore in future all the steps can be automated. We have proposed the UAS for both Android and iOS but to prove our concept, we have implemented our UAS system on Android apps only; but the sameSystem can be incorporated for the iOS applications as well.

7. References

1. Beresford AR, RiceA, SkehinN. andSohan R.MockDroid: trading privacy for application functionality on smartphones.HotMobile. 2011; p. 49-54.
2. Hornyack P, Han S, Jung J, Schechter S andWetherallD. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. ACM CCS 2011; p. 639-52.
3. Enck W, Gilbert P, Chun BG, Cox LP, Jung J, McDaniel P andSheth AN.TaintDroid: an information-flowtracking system for realtime privacy monitoring on smartphones. 9th USENIX conference on Operating systems design and implementation. 2012; p. 1-6.
4. Zhou Y, Zhang X, Jiang X and Freeh V. Taming information-stealing smartphone applications (on Android). TRUST 2011; p. 93-107.
5. Kang HS,Cho JH.Case Study on Efficient Android Programming Education using Multi Android Development Tools. Indian Journal of Science and Technology. 2015 Aug; 8(18):1-5 Doi:10.17485/ijst/2015/v8i19/75984.
6. Kang HS, Cho JH,Kim HC. Application Study on Android Application Prototyping Method using App Inventor. Indian Journal of Science and Technology. 2015 Aug; 8(18):1-5. Doi:10.17485/ijst/2015/v8i18/75919.
7. Pandey M, RajashekarBabu M, Manasa J, Avinash K. Mobile Based Home Automation and Security System. Indian Journal of Science and Technology. 2015 Jan; 8(S2):12-16Doi: 10.17485/ijst/2015/v8iS2/57792.
8. KonstantinouE and Wolthusen S. University of London:Metamorphic virus: Analysis and detection. Technical report, Information Security Group at Royal Holloway. 2009.
9. National Security Agency.SELinux.2009 January; Available from: http://www.nsa.gov/research/selinux/.
10. Chin E, Felt AP, Sekar V andWagner D.Measuring user confidence in smartphone security and privacy. In SOUPS. 2012.
11. King J. How Come I'm Allowing Strangers to Go Through My Phone?: Smart Phones and Privacy Expectations. Under review. 2012.
12. Kelley PG, Consolvo S, Cranor LF, Jung J,Sadeh N, andWetherall D. A conundrum of permissions: Installing applications on an android smartphone. In USEC Workshop. 2012.
13. Lin J, Sadeh N, Amini S, Lindqvist J, Hong JI and Zhang J. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In ACM UbiComp. 2012.
14. Book T, PridgenA, Wallach DS. LongitudinalAnalysis of Android Ad Library Permissions, arXivpreprint arXiv:1303.0857, 2013.
15. Shekhar S, Dietz M,Wallach DS.Adsplit:Separating smartphone advertising from application,CoRR. abs/1202.4030. 2013.
16. Felt AP, Ha E, Egelman S, HaneyA, Chin E and Wagner D. Android permissions: User attention,comprehension, and behavior. SOUPS 2012; p. 3.
17. Kelley PG, Consolvo S, Cranor LF, Jung J,Sadeh N and Wetherall D. A Conundrum of Permissions:Installing Applications on an Android Smartphone,Financial Cryptography and Data Security. 2012; p. 68-79.
18. Kern M and Sametinger J. Permission Tracking inAndroid, UBICOMM 2012; p. 148-55.
19. Felt AP, Ha E, Egelman S, Haney A, Chin E, Wagner D.

Android permissions: User attention, comprehension, and behavior. In SOUPS 2012.
20. Karrer K, Glaser C, Clemens C andBruder C.Technikanit aterfassen{ derFragebogen TA-EG. In 8. Berliner Werkstatt Mensch-Maschine-Systeme. 2009.
21. Helfer J and Lin T.2012 Giving the User Control overAndroid Permissions. Date accessed: 25 March 2013:Available from: http://css.csail.mit.edu/6.858/2012/projects/helferty12.pdf.
22. Berthome P, Fecherolle T, GuilloteauN andLalande JF. Repackaging Android Applications for Auditing Access to Private Data. ARES 2012; p. 388-96.
23. Juanru L, Dawu G and Yuhao L. Android MalwareForensics: Reconstruction of Malicious Events. ICDCSW2012; p. 552-58.