Permission Abusing by Ad Libraries of Smartphone Apps

Ming-Yang Su, Sheng-Sheng Chen,
Dept. of Computer Science and Information Engineering,
Ming Chuan University, Taoyuan, Taiwan
minysu@mail.mcu.edu.tw,
04360092@me.mcu.edu.tw

Tsung-Ren Wu, Hao-Sen Chang, You-Liang Liu
Dept. of Computer Science and Information Engineering,
Ming Chuan University, Taoyuan, Taiwan
ni981320ck@gmail.com, box15975316@gmail.com,
andyliu1382@gmail.com

Abstract— The security threat caused by free apps embedded with advertising has been discussing in recent years. An app developer must insert a Software Development Kit (SDK), known as ad library (ad-lib for short), into his/her host program, and compile the entire program into an Android Package (APK) executive file. This study explores the security of mobile apps from the abuse of permission made by ad-libs. Given a tested app, the proposed system in the paper may show what the ad-lib(s) embedded in the app are, and what their attempts to abuse permissions are. The proposed system used two mechanisms, *SecurityException* and *checkPermission*, to catch the attempts of permission abusing from ad-lib(s).

Keywords—app; ad-lib; permission; SecurityException; checkPermission

I. INTRODUCTION

An ad library (ad-lib, for short) is provided by the ad network, i.e., ad company, for app developers to get advertising benefits. Normally, a free app contains more than one ad-lib in order to maximize the ad benefits. The app developer needs to download ad-libs and put them into his or her host app, compile them together to become an executable APK file, which means that all of the declared permissions of the entire app, no matter from the ad-libs or from the host app, are now shared among them. If an ad-lib attempts to use a permission not declared by itself, this situation is called permission abuse.

Ad-libs caused many incidents in recent years. An Android malicious advertising virus, RottenSys, appeared in March 2018, which have infected about 5 million mobile phones [1]. Once infected, mobile phones slow down and many annoying advertisements pop out on the screen. Within 10 days from March 3 to 12, RottenSys pushed more than 13.25 million advertisements to users whose mobile phones were infected, inducing more than 540,000 advertisement clicks. In January 2018, Kaspersky Labs released an Android mobile malware called Trojan.AndroidOS.Loapi [2]. When users are browsing websites, their mobile phones would be infected once they accidentally click on the banner advertisement on the website. The Loapi virus requests the user for the administrator's privilege. If the request rejected, the prompt continues to be displayed on the screen until the user clicks the button of "confirm" before they are fed up. This study proposes a method to display the ad-libs of an app and find out the permission abuse by ad-libs. The remainder of this paper is organized as follows: Section 2 is a literature review; Section 3 is the research method; Section 4 shows some experimental results, and Section 5 is the conclusion.

This work was partially supported by the Ministry of Science and Technology with contracts MOST: 107-2221-E-130-003.

II. LITERATURE REVIEW

According to Ruiz et al. [3], the notable increase of free apps caused the top 40 ad networks to response to less than 18% of advertising request, forcing free app developers to insert more ad-libs from different companies in an app to raise the success rate of image advertising. They collected more than 625,000 apps and found that 34.88% of them have two or more inserted ad-libs. Incredibly, some apps have inserted with as many as 28 ad-libs. Stevens et al. [4] evaluated 13 ad networks and found that some issues exist in several ad-libs. The test revealed that some ad-lib uses seven undeclared permissions, including four highly violated ones: Read Calendar, Write Calendar, Read Contacts, and Write Contacts. In addition, seven of the thirteen ad-libs analyzed in that study contain JavaScript interface, meaning that these modules can execute external JavaScript.

Backes et al. [5] mentioned that free apps with built-in ads have become a trend, but apps that have been released in market or those that are in the process of development often use the ad lib of old version, resulting in potential security concerns. They designed a system to assist users in checking whether there are security concerns in the ad-libs of apps downloaded, such as obfuscated malicious behavior command. Lee et al. [6] proposed the contextual and semantic relations to distinguish app behavior from advertising behavior. Tang et al. [7] adopted static analysis of 10,710 apps, 76.08% of which were found to have permission over-claim and 424 of which include sensitive permissions that are only used in ad-libs instead of the host apps. They tackled the problems caused by the ad-libs in a semantic manner.

III. RESEARCH METHOD

Android's permission mechanism allows an app to use the corresponding hardware devices and their functions by declaring permissions in the Android Manifest.xml. If the functions required by an app without declaration, errors happen in execution and the app shuts down. Therefore, it is necessary to add a security exception mechanism to the code of an ad-lib that attempts committing permission abuse, to avoid shutdown of the app. The research method of this study is based on the assumption. When an ad-lib fails for permission abusing, in order to avoid execution errors and thus shut down the app, there are two ways for the ad-lib to deal with.

The first way is to use java.lang.*SecurityException* to catch program exceptions, i.e., the ad-lib's code that commits permission abuse is wrapped with *try catch*. When errors occur to an undeclared permission, there occurs new *SecurityException()* thrown out for exception handling. The second way is to use android.content.*ContextWrapper.checkPermission()* to confirm whether there is a permission, i.e., the ad lib uses *checkPermission()* to confirm whether to declare the permission. This study observed that this method only required the monitoring of *ContextWrapper class checkPermission()*.

This study used the Android simulator to analyze permission abusing of ad-libs. This method was to insert a test ad-lib into a blank project, called MyAPP, which does not declare any other permission than those required by the embedded ad-lib. After execution, the Android external library was monitored and the permission inquired by the ad-lib to the Android system was output. If the ad-lib only declares Permission A and Permission B, but Permission C is inquired, then Permission C is determined as permission abuse. Now Permission C was added to the Android Manifest.xml for declaration, and then continued to find other permissions abuse until no others can be found. The process is as follows:

Step 1. Create a blank project and insert a tested ad-lib: use Android to create a blank project without any declared permission. Then insert the ad-lib. Add the permissions declared by the ad-lib to Android Manifest.xml.

Step 2. Monitor *java.lang.SecurityException*, *android.content.ContextWrapper.checkPermission()* to announce the permissions declared by the advertiser. This study used breakpoints for monitoring. This study used breakpoints under *SecurityException, ContextWrapper.checkPermission()* and output the information needed in a specific format set beforehand.

Step 3. Execute the project in the simulator.

Step 4. Filter the monitored messages and store the actual permissions: Once the project is completed, the specific information got in Step 2 is retrieved.

Step 5. The differences set between the actual permissions and the advertiser's declared permissions helps to obtain the permission abusing. If it exist, add this abused permission to the Android Manifest.xml. Return to Step 3 until no new one can be found.

In order to find out all permission abuses, once a permission abuse explored, it should be added to AndroidManifest.xml, and then return to Step 3 to find the next one.

IV. EXPERIMENTAL RESULTS

This study included 26 ad-libs. Nearly 90% of the apps were allegedly [8] inserted with at least one of the 26 ad-libs. By the experiment, the result is shown in Table 1, the declared permissions were marked as O and the undeclared but inquired permissions are marked as X. Most of the ad-libs have undeclared but inquired permissions, like MODIFY AUDIO SETTINGS and BLUETOOTH. For examples, the ad-lib G abused permission READ CONTACTS, while ad-lib E and F abused permission ACCESS_COARSE_LOCATION.

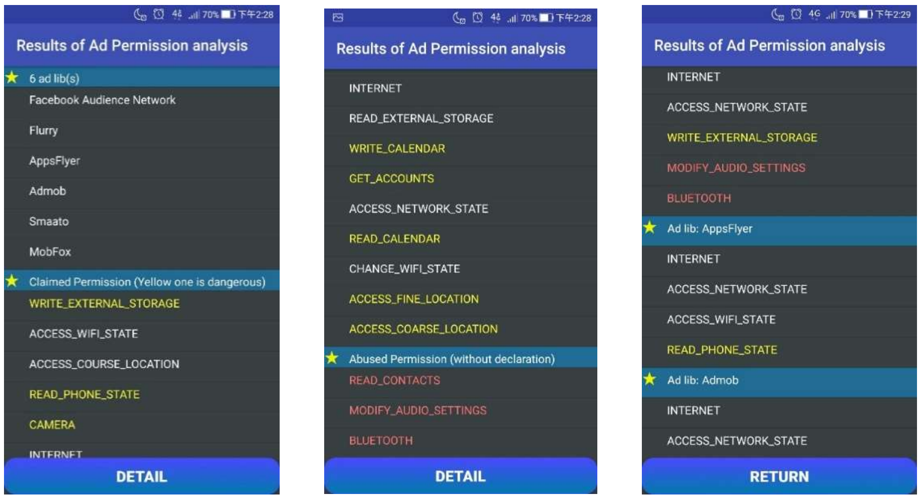
The proposed system checks Table 1 as reference after de-compiling the APK and getting the names of as-libs, then displays the users the information: what ad-libs are embedded and what permissions are used or even abused by ad-libs in the APK. Figure 1(a) showed that six ad-libs (upper in the figure) are embedded in the tested APK and their declared permissions (middle). Figure 1(b) showed the abused permissions in red (lower). Figure 1(c) showed how the permissions of individual ad-libs, such as AppsFlyer, are used (middle). The yellow permissions in Figure 1 referred to dangerous ones according to the official Android website [9].

V. CONCLUSION

According to this study, there seems to be a large proportion of ad-libs committing permission abuse, but fortunately, what involved are mostly not sensitive permissions. Some ad-libs that do not commit permission abuse may have declared excessive permissions that are beyond the needs of advertising. The proposed system in this paper enables users to know whether an app has attempts to access undeclared permissions. In doing so, users could be aware of the danger of apps, so that most people are encouraged to use highly reputable applications (including embedded ad-libs) for a virtuous circle.

TABLE I. LIST of AD-LIBS ATTEMPTING PERMISSION ABUSE

Per. / Ad-Lib	INTERNET	ACCESS_NETWORK_STATE	READ_PHONE_STATE	CALL_PHONE	WRITE_EXTERNAL_STORAGE	READ_CALENDAR	WRITE_CALENDAR	READ_CONTACTS	RECEIVE_BOOT_COMPLETED	GET_ACCOUNTS	ACCESS_WIFI_STATE	ACCESS_FINE_LOCATION	ACCESS_COARSE_LOCATION	CHANGE_WIFI_STATE	INSTALL_SHORTCUT	VIBRATE	MODIFY_AUDIO_SETTINGS	BLUETOOTH	READ_EXTERNAL_STORAGE	PACKAGE_USAGE_STATS	BLUETOOTH_ADMIN	GET_TASKS	REAL_GET_TASKS	NFC	RECORD_AUDIO	ACCESS_COARSE_LOCATION	USE_CREDENTIALS	WAKE_LOCK	SYSTEM_ALERT_WINDOW
A	O	O	O								O																		
B	O	O			O												X	X											
C	O	O	O								O																		
D	O	O	O		O														O										
E	O	O			O						O	X	X				X	X											
F	O	O			O						O		X			O	X	O	O					O	O				
G	O	O	O		O			X				O	O	O			X	X								O			
H	O	O			O							O	O				X	X											
I	O	O	O																										
J	O	O	O		O	O	O			O		O	O				X	X											


Results of Ad Permission analysis

6 ad lib(s)
Facebook Audience Network
Flurry
AppsFlyer
Admob
Smaato
MobFox
Claimed Permission (Yellow one is dangerous)
WRITE_EXTERNAL_STORAGE
ACCESS_WIFI_STATE
ACCESS_COURSE_LOCATION
READ_PHONE_STATE
CAMERA
INTERNET

DETAIL

Results of Ad Permission analysis

INTERNET
READ_EXTERNAL_STORAGE
WRITE_CALENDAR
GET_ACCOUNTS
ACCESS_NETWORK_STATE
READ_CALENDAR
CHANGE_WIFI_STATE
ACCESS_FINE_LOCATION
ACCESS_COARSE_LOCATION
Abused Permission (without declaration)
READ_CONTACTS
MODIFY_AUDIO_SETTINGS
BLUETOOTH

DETAIL

Results of Ad Permission analysis

INTERNET
ACCESS_NETWORK_STATE
WRITE_EXTERNAL_STORAGE
MODIFY_AUDIO_SETTINGS
BLUETOOTH
Ad lib: AppsFlyer
INTERNET
ACCESS_NETWORK_STATE
ACCESS_WIFI_STATE
READ_PHONE_STATE
Ad lib: Admob
INTERNET
ACCESS_NETWORK_STATE

RETURN

(a) All ad-libs and declared permissions (b) Permission abusing (c) Individual ad-lib's permissions

Fig 1. Example: An APK inserted with six ad-libs and their permissions attempts

REFERENCES

[1] Pre-Installed Malware Found On 5 Million Popular Android Phones, March 2018.
https://thehackernews.com/2018/03/android-botnet-malware.html

[2] Lo lo lo Loapi Trojan could break your Android.
https://blog.malwarebytes.com/cybercrime/2017/12/lo-lo-lo-lo-loapi-we-have-you-protected/

[3] Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst, and Ahmed E. Hassan, "Impact of Ad Libraries on Ratings of Android Mobile Apps," IEEE Software, Vol. 31, Issue 6, pp. 86-92, 2014.

[4] Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Ericksonand, and Hao Chen, "Investigating User Privacy in Android Ad Libraries," in the Proceedings of the IEEE Mobile Security Technologies (MoST), San Francisco, CA, May 24, 2012.

[5] Michael Backes, Sven Bugiel, and Erik Derr, "Reliable Third-Party Library Detection in Android and its Security Applications," in the Proceedings of the 23rd ACM Conference on Computer and Communications Security, 2016.

[6] Jung-Hyun Lee, So-Young Jun, So-Jung Park, Kang-Min Kim, SangKeun Lee, "Demo: Mobile Contextual Advertising Platform based on Tiny Text Intelligence," in the Proceedings of the 15th ACM International Conference on Mobile Systems, Applications, and Services, 2017.

[7] Junwei Tang, Ruixuan Li, Hongmu Han, Heng Zhang, Xiwu Gu, "Detecting Permission Over-claim of Android Applications with Static and Semantic Analysis Approach," in the Proceedings of the IEEE Trustcom/BigDataSE/ICESS, Sep. 2017.

[8] AppBrain, Android Ad networks, December 2018, http://www.appbrain.com/stats/libraries/ad

[9] Permissions overview,https://developer.android.com/guide/topics/permissions/overview#normal-danger