Analyzing the Installation logs of an Android Application

Sumin Lee
Soongsil University
Seoul, Korea
smlee.oslab@gmail.com

Minho Park
Soongsil University
Seoul, Korea
mhpark.oslab@gmail.com

Kijun Kim
Soongsil University
Seoul, Korea
deukrayx@gmail.com

Danial Hooshyar
Tallinn University
Tallinn, Estonia
danial@tlu.ee

Jiman Hong*
Soongsil University
Seoul, Korea
jiman@ssu.ac.kr

ABSTRACT

Malicious applications not only directly perform malicious actions, also have a function of inducing them to install separate malicious applications. In particular, if an application is installed through a private application store or black market, it is known that the possibility of a malicious application is high. In this paper, we propose a method to analyze the installation log of an Android app. The proposed method checks the installation log by comparing the log file before app installation and the log file created or modified inside the Android system after the app installation.

KEYWORDS

Android, Log File, Malware, App Store, Black Market

1 INTRODUCTION

Android Mobile Platform is the most used in the world. According to Statcounter, even Windows lags behind the widespread adoption of the functional smartphone platform owned by Google and still powers about 71.8% of all smartphones and tablets[1]. Android apps can be installed through the official Google Play Store[2] or through private app stores and black markets. While Google Play Store performs security checks on uploaded apps, if you install apps through private app stores and black markets that do not provide these checks, you may be more vulnerable to leakage of users' personal information due to malicious apps or malicious codes.

Various studies have been conducted to analyze Android malicious apps or malicious codes. In [3], Hong et al. studied an efficient user behavior-based live evidence collection and analysis system using the Android log system. After determining the log pattern that can identify the user's behavior through pre-profiling, a method for constructing a database by expressing the pattern as a regular search expression was proposed. In addition, unnecessary logs were removed through the purification process. Combining multiple log entries together can increase the accuracy of the analysis.

In [4], Vidas et al. studied a collection method that can be universally applied to Android devices. The boot image is taken as an example of Android internal information, and after checking the boot mode of a special device and the partition schema of Android, the specific combination of the Android boot image is analyzed. By reflecting the analyzed contents, the form of a boot image that can be usefully used for forensic data collection was proposed. If the proposed design is applied to a real device, it can be utilized for boot image forensic analysis.

In [5], Votipka et al. implemented Passe-Partout, a collection method that can be universally applied to Android devices. Passe-Partout can be applied to platforms other than Android, it can have the advantage of collecting necessary forensic data without additional procedures or illegal procedures such as Android rooting.

However, in previous studies, the main research method was to ignore the installation method and analyze all apps for malicious code. If the analysis target of vulnerable Android apps is analyzed for malicious code in apps downloaded through private app stores or black markets, excluding apps downloaded from official stores that have undergone security checks by default, the number of apps to be analyzed can be greatly reduced.

Therefore, in this paper, we propose a method to analyze the installation logs of an Android app. Many log files are created in the Android system during a short time when an app is installed on a mobile device with Android. This log file contains the package name of the installed app, the installation method and time, and update information. The proposed method analyzes only the logs created or modified when installing Android apps and classifies apps installed through private app stores and the black market other than Google Play Store. Since the proposed method does not analyze all log files, it can reduce the time to analyze the installation logs of the installed app.

The remainder of the paper is organized as follows. Section 2 describes how to extract the Android app installation logs and Section 3 describes the log created or modified for each Android app installation method. Section 4 presents the log files to identify the installation method. Finally, concluding remarks are provided in Section 5.

* Corresponding Author

2 Extracting Android App Installation Logs

To analyze the log file of the Android app installed on the mobile device, it is necessary to compare the log file created or modified before and after the target app is installed. In general, after the Android app is installed, all data files related to the app exist in '/data/data', as shown in Fig. 1, all folders and files existing under the folder should be copied to the designated folder of the Host PC to collect log files. When the copy is complete, the installed package name, installation time, and installation log information are extracted from the log file through log file analysis.

```
procedure Dumping(dirPath) :
  # copy /data/data to host computer.
  cpCmd = 'adb shell su -c cp -R /data/data ' + dirPath
  os.system(cpCmd)
  # compress directory into tar
  tarCmd = 'adb shell su -c tar cf ' + tarName + ' ' +
    dirPath
  os.system(tarCmd)
  # pull tar file into current directory from Android
    device.
  pullCmd = 'adb pull ' + tarName + ' ./' + dirPath
  os.system(pullCmd)
end
```

Figure 1: Copy directories and files under '/data/data' to Host PC

To identify the modified file, we use the Linux meld program that finds the changed file inside the folder. The meld program takes two directories as input, provides sub-directories and file comparison, and finds the modified files. If there is no extension in the log file or if the file extensions are not mainly used such as '.temp', it is not possible to compare modified files. Therefore, if it is impossible to check the file contents using the meld program, programs such as *strings*, *xxd*, *od* are used on the terminal, and in the case of the db file, use *sqlitebrowser* to check the changed contents.

3 Android App Installation Logs Analysis

In '/data/data', information of all apps installed on Android mobile devices is stored. Therefore, in order to efficiently analyze the log files that are created or modified when installing the app, the app's log file is analyzed by targeting the Google Play Store app internal storage folder. When the APK downloaded from the official app store, Google Play Store, and the unofficial app store, is installed using the adb command, the files in the Google Play Store app internal storage folder are modified. Details are described in 3.1 and 3.2.

3.1 App installation using Google Play Store

When an app downloaded from Google Play Store is installed on a mobile device, the 'scheduler_main', 'localappstate.db', 'library.db', and frosting.db' files in the databases folder of the Google Play Store app internal storage folder are modified.

The 'scheduler_main' file has a total of 3 tables: 'jobs', 'android_metadata', and 'jobs_audit'. After installing the app, a record is added only to the 'jobs' table, but information about the installation method is not recorded.

There are two tables 'android_metadata' and 'appstate' in the 'localappstate.db' file. After the app is installed, there are no records added to the 'android_metadata' table, but the appstate table stores the package name and installation-related information for the installed app. In the 'install_reason' property of the 'appstate' table, when installing through the Google Play Store, it is recorded as 'single_install', and if the app is installed through a different path, the database is not changed.

The 'library.db' file contains two tables, 'ownership' and 'android_metadata'. After installing the app, a record is added only to the 'ownership' table. In the added record, the package name is recorded and 'purchase_time' similar to the app installation time is recorded, but the installation method is not recorded.

The 'frosting.db' file consists of tables of 'frosting', 'frosting_audit', and 'android_metadata', and records are added only to the 'frosting' table. The value of the 'last_updated' attribute of the 'frosting' table is changed when an app is newly installed or an update is in progress.

3.2 App installation using adb command

'scheduler_main', 'verify_apps.db', and 'frosting.db' existing in the databases folder of the Google Play Store app internal storage folder are modified. In the case of installation through the adb command, the 'localappstate.db' file is not changed.

Records are added to the 'apk_info' and 'installation_attempts' tables of the 'verify_apps.db' file, and no new records are added to other tables. The content added to the 'apk_info' table has a 'pk' attribute and a 'data' attribute. The 'pk' attribute is the SHA256 hash value of the APK file, and the 'data' attribute stores information such as the SHA256 value of the APK file, the package name of the installed app, and the name of the installed app.

4 EVALUATION

To evaluate the proposed method properly extracts the log files created or modified according to the app installation method, rooted Google's Pixel 4a5G was used and 4 of the most downloaded apps [5] worldwide in 2021 selected by Apptopia were used. Table 1 shows the names and versions of the apps used in the experiment.

To compare log files that are created or modified according to the app installation method, install the APK downloaded through the Google Play Store, the official store, on the rooted Google Pixel 4a5G, and delete the APK after extracting all logs in the Google Play Store app internal storage. Then, install the APK downloaded from APKPure[6], an unofficial app store, using the adb command, and extract the logs.

Table 1: Version of Target Apps

Target App	Version
Facebook	380.0.0.29.109
Instagram	248.0.0.17.109
WhatsApp	2.22.18.17
Spotify	8.7.56.421

Table 2 shows the results of comparing log files modified when apps downloaded from the Google Play Store and unofficial app stores are installed, respectively. The case where the existing log file is modified is indicated by 'O', and the case where there is no modification is indicated by 'X'. The Google Play Store records the presence of updates and how they are updated (automatic or manual). In addition, information such as the installation time and update time of the currently installed app is recorded. Therefore, through the internal storage files of the Google Play Store app, databases/localappstate.db and databases/verify_apps.db, an app installed through the Google Play Store and an app installed through a different path can be distinguished.

Table 2: List of files modified according to the installation

File	Google Play Store	adb
databases/localappstate.db	○	×
databases/scheduler_main	○	○
databases/verify_apps.db	×	○
databases/library.db	○	×
databases/frosting.db	○	○

5 CONCLUSIONS

In this paper, we proposed a method to analyze the installation logs of an Android app. In order to verify the validity of the proposed method, we collected and analyzed the installation logs recorded in a file in the Google Play Store app internal storage using a rooted Pixel 4a5G. As a result of analyzing the log that stores information such as package name, installation time, update time, and installation method when installing an app among the logs of the Google Play Store app, it was confirmed that apps installed through the Google Play Store and apps installed in a different path can be distinguished.

It is expected that the file created or modified by the installation method can be used as analysis or evidence for those in the field of data forensics. However, in this study, only the installation logs related to the Google Play Store and apps installed through the adb command were analyzed. Since the Android system can proceed with APK installation in various ways, various analyses will be possible if you check the internal storage files of several basic packages including the corresponding package.

ACKNOWLEDGMENTS

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the National Program for Excellence in SW)(2018-0-00209) supervised by the IITP(Institute of Information & communications Technology Planning &Evaluation)

REFERENCES

[1] Statcounter, Operating System Market Share Worldwide, https:// https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202207 (accessed Aug., 18, 2022)
[2] Ilyoung Hong, Sangjin Lee, “Research on Efficient Live Evidence Analysis System Based on User Activity Using Android Logging System”, Journal of The Korea Institute of Information Security and Cryptology, vol. 22, no. 1, pp. 67-80, 2012.
[3] Timothy Vidas, Chengye Zhang, Nicolas Christin, “Toward a general collection methodology for Android devices”, Digital Investigation: The International Journal of Digital Forensics & Incident Response, vol. 8, pp. S14-S24, 2011.
[4] Daniel Votipka, Timothy Vidas, Nicolas Christin, “Passe-Partout: A General Collection Methodology for Android Devices”, in IEEE Transactions on Information Forensics and Security, 2013.
[5] Worldwide and US Download Leaders 2021, https://blog.apptopia.com/worldwide-and-us-download-leaders-2021 (accessed Aug., 18, 2022)
[6] APKPure, https://apkpure.com/in/ (accessed Aug., 18, 2022)