Protecting Data in Android External Data Storage

Hao Zhang, Zhuolin Li, Hossain Shahriar, Dan Lo
College of CSE
Kennesaw State University
{hzhang13, zli29}@students.kennesaw.edu
{hshahriar, dlo2}@kennesaw.edu

Fan Wu
Dept. of Computer Science
Tuskegee University
fwu@tuskegee.edu

Ying Qian
Dept. of Computer Science
East China Normal University
yqian@cs.esnu.edu.cn

***Abstract*—Insecure data storage may open a door to malicious malware to steal users' and system sensitive information. These problems may due to developer negligence or lack of security knowledge. Android developers use various storage methods to store data. However, Attackers have attacked these vulnerable data storage. Although the developers have modified the apps after knowing the vulnerability, the user's personal information has been leaked and caused serious consequences. As a result, instead of patching and fixing the vulnerability, we should conduct proactive control for secure Android data storage. In this paper, we analyzed Android external storage vulnerability and discussed the prevention solutions to prevent sensitive information in external storage from disclosure.**

Keywords—external storage, proactive control, encryption, permission, monitoring, vulnerability

I. Introduction

Obviously, the Android market is growing very fast, but everything that's going forward at such a rapid pace inevitably comes back with some risks, like data leaking. Because there are too many Google play store apps, it is difficult to strictly monitor each application, and the approval rate is very high. At this time, malicious Android app developers apply for approval with a fluke mentality, which will bring risks to users.

Insecure data storage is a common issue for mobile devices. The Open Web Application Security Project(OWASP) lists insecure data storage is one of the top 10 security issues in mobile phones [1]. If the data storage is not well protected, malicious malware and attackers can easily access personal or corporate sensitive data stored on the device. Insecure data vulnerability often leads to some risks such as fraud, identity theft, reputation damage, and material loss. However, this security weakness is very common, the reason is that mobile app developers sometimes think that users or hackers will not have access to a mobile device's data storage. What's more, many development professionals lack awareness of the importance of security vulnerabilities and the necessary security knowledge and skills when they developing apps.

Android provides a variety of options for storing data: shared preferences, SQLite databases, internal storage and external storage. Shared preferences are XML files to store private data in key-value pairs. SQLite databases are lightweight file-based databases which Android can support. Internal storage is another method we store our data. It will save users' data on the device. External storage is a location out of the device, and it usually removable [2]. In this paper, we focus on external storage.

External storage is a removable location that user can save data. The purpose of external storage is to add extra storage for the mobile phone. The most common example of external storage is SD card.

The security of mobile applications has been seriously threatened, and it is vital to find ways to protect sensitive data. Proactive control for secure Android data storage is necessary in case of a malicious attack. These proactive controls help software developers consistently and thoroughly protect users' data throughout the application development process, rather than patching and fixing bugs.

In order to prevent users' sensitive information from being leaked, we put forward four suggestions. (1) Avoid storing important and sensitive information in external storage. (2) Add monitoring technology and log monitoring information. So the hackers will be recorded. (3) Encrypting the data so that hackers can't easily get the user's information even if they have access to it. (4) Add permission, so those people who without permission cannot access into user's data.

When processing data from external storage, input validation should be performed as if it were from any untrusted sources. Executable or class files should not be stored on external storage until dynamically loaded. If your application does retrieve executable files from external storage, the files should be signed and encrypted for verification before dynamic loading.

II. Related Works

Some researchers have explored different Android vulnerability about data storage. Altuwaijri, H., and S. Ghouzal 2018[3] investigated the security of the Android data storage model from 2013 to 2018. They found several threats that can be classified as physical or software threats. While Android provides valuable encryption systems, including full disk encryption and key chains, to enhance the security of data stores, encryption keys stored on devices are still vulnerable to physical threats.

III. Vulnerability Analysis

First, we look at external storage vulnerability. Whatsapp is a classic example of SD card storage. This app stores all the data on the SD card [4]. This may lead to insecure data risk because hackers can remove users SD card and plug it to a new device. The data will fully accessible to hackers.

978-1-7281-2607-4/19/$31.00 ©2019 IEEE
DOI 10.1109/COMPSAC.2019.00143

IEEE computer society

Because Google Play has a high approval rate for apps, many malicious Android apps can be downloaded on Google play. When a user downloads the malicious app, this malicious app can easily steal the user's data if the user does not set any permission on the device.

We made a simple app called "External Storage test" on our virtual device. The data storage location of this app is in the SD card. Firstly, we put our test data on the virtual device. As shown in the left part of Fig. , and we input "This is the test for external storage!" and click the "Save" button to save the data into the SD card. We have saved the data entered by the user into a txt file with the name "external_storage.txt."

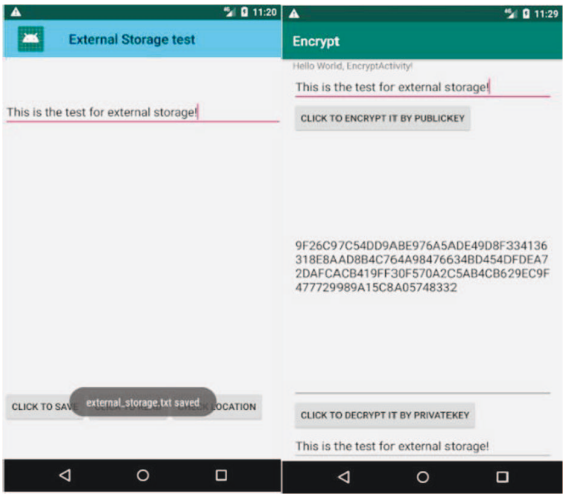

Figure.1 Original data and encrypted data

```
1312313(base) hao@hao-N551JW:~/Downloads$ adb pull /sdcard/external_storage.txt
[100%] /sdcard/external_storage.txt
(base) hao@hao-N551JW:~/Downloads$ cat external_storage.txt
This is the test for external storage!(base) hao@hao-N551JW:~/Downloads$
```

Figure.2 check external vulnerability

In Fig. 2, we use Android Debug Bridge pull command to download 'external_storage.txt" from our mobile emulator to our local computer. Then, we use cat command in Linux to concatenate file and print on the standard output in the terminal. As we can see, if anyone has physical access to the storage, the data entered by the user will be easily accessible by the attacker. For example, a malicious app which a user download from Google play can use Android Debug Bridge to obtain the user's information.

IV. Data Protection Solutions

In order to prevent the user's sensitive data from being leaked, we proposed four methods to protect the data. (1) Avoid storing important and sensitive information in external storage. Since external storage is publicly accessible to anyone, we should try to avoid storing sensitive information in external storage. (2)Monitoring and logging all external data storage access activities with *logcat* and relevant API event handlers. With monitoring and logging all suspicious actions will be recorded. (3) Encrypting the data at rest in external data storage. As a result, a hacker cannot easily obtain information about a user, even if the hacker has physical access to the user's data. (4) Add permission to read and write external data, so those people who without physical permission cannot access into SD card storage.

Event handling can record any actions which perform in the application and save these data in a log so that it can act as a monitor. So the user can detect the hacker's behavior.

In Fig. 3, we made a simple app to encrypt the data with RAS. RSA is a popular cryptophytic method for encoding and decoding the data in data storage

```
private static byte[] encrypt(String text, PublicKey pubRSA)
        throws Exception {
    Cipher cipher = Cipher.getInstance(RSA);
    cipher.init(Cipher.ENCRYPT_MODE, pubRSA);
    return cipher.doFinal(text.getBytes());
}
public final static String encrypt(String text) {
    try {
        return byte2hex(encrypt(text, uk));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Figure 3. RSA encryption fragment

In the right part of Fig. 1, we click the button to encrypt a string data "This is the test for external storage". In this way, even if the hackers gets the information, they will only see the encrypted information and the user's data will be protected. When clicking "decrypt" button, the encrypted data will become original data.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Figure. 4 Permission code fragment

In order to protect users' data from being leaked, we should set permissions. Fig. 4 shows how to set read and write permissions for external data access.

V. Conclusion

We discussed proactive controls for secure Android external data storage. With data encryption method is explained in detail even if the encrypted data is accessed by an intruder, the contents stored by the user cannot be viewed. Developers should strengthen the protection of user data to prevent malicious intrusion. The mobile external data protection solutions are also discussed.

ACKNOWLEDGEMENTS

The work is partially supported by the U.S. National Science Foundation under awards: NSF proposal 1723586, 1723578, 1636995, and 1623724, Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation and Department of Homeland Security.

References

[1] "OWASP Mobile Security Project - OWASP", 2016, *Owasp.org,* https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks.

[2] "Insecure Local Storage: Shared Preferences". *Infosec,* https://resources.infosecinstitute.com/android-hacking-security-part-9-insecure-local-storage-shared-preferences/#article.

[3] Altuwaijri, H., and S. Ghouzali. "Android Data Storage Security: A Review." 2018, *Journal of King Saud University* - Computer and Information Sciences.

[4] "Insecure Local Storage". *Infosec,* https://resources.infosecinstitute.com/android-hacking-security-part-10-insecure-local-storage/#article