Privacy Leaks through Data Hijacking Attack on Mobile Systems

Daojuan Zhang[1,2], Yuanfang Guo[1], Dianjie Guo[1] and Guangming Yu[3]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, 100093, China*
[2]*University of Chinese Academy of Sciences, Beijing, 100049, China*
[3]*Chinese Research Institute of General Technology, Beijing, 100055, China*
Email: {zhangdaojuan, guoyuanfang, guodianjie}@iie.ac.cn, yuguangming2010@163.com

Abstract—To persistently eavesdrop on the mobile devices, attackers may obtain the elevated privilege and inject malicious modules into the user devices. Unfortunately, the attackers may not be able to obtain the privilege for a long period of time since the exploitable vulnerabilities may be fixed or the malware may be removed. In this paper, we propose a new data hijacking attack for the mobile apps. By employing the proposed method, the attackers are only required to obtain the root privilege of the user devices once, and they can persistently eavesdrop without any change to the original device. Specifically, we design a new approach to construct a shadow system by hijacking user data files. In the shadow system, attackers possess the identical abilities to the victims. For instance, if a victim has logged into the email app, the attacker can also access the email server in the shadow system without authentication in a long period of time. Without re-authentication of the app, it is difficult for victims to notice the intrusion since the whole eavesdropping is performed on other devices (rather than the user devices). In our experiments, we evaluate the effectiveness of the proposed attack and the result demonstrates that even the Android apps released by the top developers cannot resist this attack. Finally, we discuss some approaches to defend the proposed attack.

I. Introduction

Nowadays, the mobile device is extremely popular and has become a primary choice to store and handle the privacy data, e.g., SMS messages, contact information, etc. Unfortunately, the mobile devices also attract more attacks and the safety of these privacy data has received lots of concerns [11, 8, 13, 12, 14]. To steal the user privacy persistently, one of the common methods for the attackers is to obtain the elevated privilege of the user devices and inject the eavesdropping modules. During this process, the users are usually lured to install the malwares from the attackers, while some users become confused and grant the elevated privilege to the attackers. In addition, attackers can directly obtain the elevated privilege by exploiting the system vulnerabilities. Some spies can even access the devices of victims physically and then inject the eavesdropping modules. However, there are some limitations for the attackers. For example, they may only obtain the privilege temporarily because the malware is found and deleted by the user. Besides, by injecting the malicious modules into the user devices, it is easy for the attackers to expose themselves.

In this paper, we propose a new data hijacking attack for Android apps. By only obtaining the elevated privilege of the user device once, a long-time eavesdropping can be achieved by the attackers. The proposed attack is based on three facts. Firstly, mobile apps usually authenticate the users with passwords at the first time for convenience. The session keys are usually stored in data files for the future authentication. Secondly, data files of the mobile apps are jointly managed by the OS and can be located easily. Thirdly, for an account, multiple connections are usually allowed by the app servers at the same time, and these different connections cannot be distinguished whether they are from the same device. Thus, once the attackers obtain the elevated privilege, they can hijack the data files of users to construct a similar execution environment, named shadow system in this paper. If the users have logged into their apps at the attack moment, attackers will also be able to access the app server in the shadow system without authentication. As a result, the attack is still effective in the shadow system even if the elevated privilege of the user device expires. Besides, without re-authentication, it is difficult for the users to find the malicious behavior since the whole eavesdropping process is implemented on the

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

device of the attackers rather than the users. To evaluate the effectiveness of the proposed attack, we design a generic construction of the shadow system for Android apps and conduct the attack with 7 most popular real-world apps released by the top developers. The results show that the proposed attack can be conducted successfully even on Gmail, which are popular and downloaded more than one billion times.

In summary, our contributions of this paper are as follows.

- We propose a new data hijacking attack for mobile apps. By constructing the shadow system, we only need to obtain the elevated privilege once and can persistently steal the user privacies without any change to the user devices.
- We evaluate the effectiveness of the proposed attack by constructing a shadow system. The experimental results on the most popular Android apps indicate that most of the apps cannot resist the proposed attack.
- After the demonstrations of the proposed attack, we give some possible solutions to mitigate the data hijacking attack.

The rest of the paper is organized as follows. Section 2 introduces our motivation of this paper. The detailed data hijacking attack is described in Section 3. Section 4 presents the evaluation of the proposed attack on the popular Android apps. Solutions to resist the attack are discussed in Section 5. Finally, we conclude this paper.

2 Motivation

Since the security issues about the mobile devices are exposed frequently, we should be aware that mobile systems are not secure, especially for the Android system. According to a recent study [22], 73 Android vulnerabilities are collected and all of them can be exploited by the attackers to obtain the root privilege of the user devices. Some other researches such as [23] also reveal the severity of this problem that these vulnerabilities are exploited to achieve different activities by many malwares. Besides, even the regular app developers [1, 2, 3] exploit these vulnerabilities to help users to achieve a complete control of their devices. Once the attackers obtain the elevated privilege, numerous serious problems can be caused.

Since the system vulnerabilities may be fixed and the malware may be deleted, the elevated privilege may be obtained temporarily by the attackers. Besides, some spies can access the devices of victims physically, and do not intend to perform any change to the devices such that they can avoid being exposed. Thus, the user privacies on mobile devices can be stolen in a short period of time, but they are difficult to be eavesdropped persistently.

In this paper, a new data hijacking attack is proposed, and a shadow system is designed to steal the user privacies persistently and quietly. Via the attack practice, we hope to remind the app developers to design proper security mechanisms for protecting the users' data in an insecure execution environment, and also warn the users of the severe risks which can be caused by the abuse of elevated privilege.

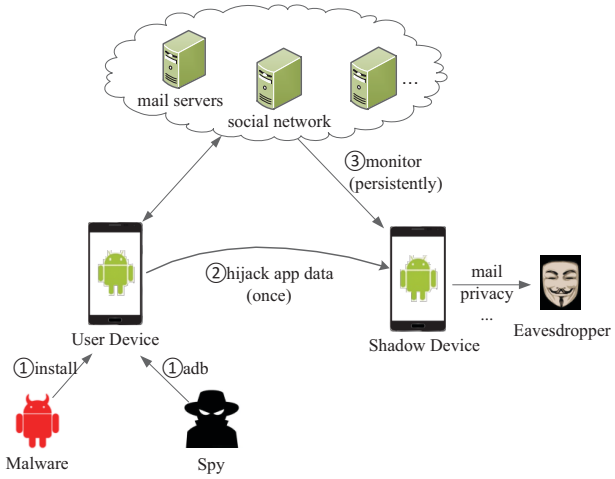

Figure 1. The process of the data hijacking attack.

3 The Proposed Data Hijacking Attack

3.1 Attack Process

As shown in Figure 1, the proposed data hijacking attack mainly consists of three steps.

Firstly, the attackers need to obtain the elevated privilege in the user devices. We assume that the privilege is obtained when the malwares are installed in the user devices or the devices are temporarily controlled by a spy. These approaches [22, 1, 2, 3, 4] can be employed to obtain the elevated privilege in the user devices.

Then, the attackers hijack the app data files in the user devices to construct the shadow system. In this phase, according to the information of the user device, a similar system (shadow system) is built and the same apps are installed by attackers. Besides, the app data files in the shadow system are replaced by the corresponding hijacked files.

Finally, in the shadow system, the attackers will have the same abilities as the victims at the attack moment. For example, if the users have logged in an app, the attackers can also access the app servers to view the user privacy such as email without authentication.

3.2 App Data Storage in Android

In order to implement the data hijacking attack, we firstly analyze the data storage of the app in the mobile system. Take Android as an example, the three critical locations for storing app data files or account data are as follows.

Private App Data. It is well-known that Android is constructed based on the Linux kernel. To isolate the apps, Android system creates different system users for the

TABLE I. The evaluation results on popular Android apps.

ID	Package Name	Category	Attack Requirements & Existing Defenses			Duration of Attack
			Unique ID	System	Protocol	
1	com.tencent.mobileqq	Comm	-	-	Single User Login	Fail in the first week
2	com.baidu.netdisk	Cloud	-	-	-	Persistent
3	com.sina.weibolite	Social	-	-	-	Persistent
4	com.netease.mail	Mail	-	-	-	Persistent
5	com.google.android.gm	Mail	-	Sync	-	Persistent
6	com.facebook.katana	Social	-	-	SMS verification	Incomplete verification
7	com.microsoft.skydrive	Cloud	-	Sync		Fail in the first week

installed apps with unique UID. For each app, there is a private directory in the system directory /data/data/ to store its private data files, which cannot be accessed by other unprivileged apps.

External Storage. Besides the private directory, the app data can be also stored in the external storage such as SDCard by an Android app. Files in the external storage can be accessed by all apps who have the permission *READ_ EXTERNAL_STORAGE*. It is obvious that the critical data cannot be stored in external storages by the developers. In our experiment, this part of app data is not concerned.

Sync Adapter. The Sync Adapter [5] is a service provided by the Android system to transfer the data between the client and the server. Apps can store their account or session information based on this service. Thus, app data such as user contacts can be regularly synchronized with the cloud servers. The account information is stored in the system file */data/system/users/0/accounts.db*, which is a database file with SQLite format.

3.3 Differential Data Analysis

Although it is effective to directly hijack all the app data files into the shadow system, we introduce two differential data analyses to reduce the size of required files.

Firstly, when an app is installed in the different execution environments, many files such as the codes and resource files are identical. Assuming that the app P is installed in the device D, its data files are denoted as $F = \{f_1, f_2, ...\}$. Instead of hijacking all the files in F to the shadow system D', we install app P in D' and execute it to generate the files $F' = \{f'_1, f'_2, ...\}$. By computing the file signatures, only the different files $(F-F \cap F')$ are required to be hijacked. Thus, the size of the hijacked files can be reduced significantly.

Aiming at specific apps, the files related to the user account information can be located more accurately. Before attacking an app P, a legal account is registered in advance. And the files generated by the app before and after the user logging into the account are denoted as F_1 and F_2, respectively. Thus, the changed files $(F_2 - F_1)$ are related to the user account. However, although fewer files need to be hijacked by this approach, the apps may not be normally executed due to the lack of some critical files.

4 Evaluation

4.1 Environment Setup

We simulate the proposed attack with a real Android device and the corresponding shadow system is constructed by the Android emulator. The real device we used is Samsung Galaxy S6 with Android OS 5.1.1. Firstly, Android apps are installed in both the real device and the shadow system. Then, the account is registered and logged into the app in the real device. Finally, the different data files in the real device are copied to replace that in the shadow system.

Some aspects need to be noticed. When the data files are hijacked from the real devices to the shadow system, their permissions must be reset to ensure that the apps in the shadow system can be successfully executed. In our experiment, this process is automatically completed by a script program. Besides, since some account information of apps is stored into the SQLite database file (*accounts.db*) which may not be compatible for different systems, we write a JDBC program to transfer the data.

4.2 Experimental Results

As shown in Table I, 7 most popular Android apps released by the top developers are exploited to evaluate the effectiveness of the proposed hijacking attack. Our evaluation addresses the following two research questions:

- **RQ1** Can the apps discover the data hijacking attack and stop their services to the attackers?
- **RQ2** How long does the attack last if the app is attacked?

The results indicate that all the evaluated apps can be attacked. This experiment indicates that the user privacies can be accessed by the attackers in the shadow system without authentication. The QQ (*com.tencent.mobileqq*) is the most popular instant messaging app in China. Once the app is attacked, chat records can be accessed by the attackers. Thankfully, the app does not allow multiple users login to the app at the same time. If the attacker login the same account when the user is online, the attacker will be exposed. The Gmail (*com.google.android.gm*) and Microsoft OneDrive (*com.microsoft.skydrive*) manage their account connections by the Sync Adapter mechanism. To perform attack on these two apps successfully, the data in the system database file */data/system/users/0/accounts.db* needs to be hijacked. The left apps, such as *com.sina.weibolite* and *com.netease.mail*, can be simply attacked just by hijacking their app files. We persistently trace the user information for a month and only three apps re-authenticate their users. The QQ and Microsoft OneDrive correctly perform authentication and the attack fails in the first week, while the process of Facebook (*com.facebook.katana*) is incomplete. The Facebook server re-authenticates its user by sending the SMS verification code. If the code is not verified, both the user and the attacker cannot login to the app. However, once the user inputs the code, the attackers can also connect the app server in the shadow system again without authentication.

4.3 Case Study

Gmail is an email app developed by Google and downloaded more than 1 billion times. Figure 2 presents the result when the app data files of Gmail are hijacked. Gmail updates the email information based on the Sync Adapter mechanism. By replacing the system database *account.db* in the shadow system, our Gmail testing account can be logged in without authentication. The execution of Gmail depends on some other Google services such as Google Play Service and Google Framework Service. However, these services can be directly installed in the shadow system and the attack is not affected. In our experiment, the emails in the shadow system can be read persistently for a month without authentication.

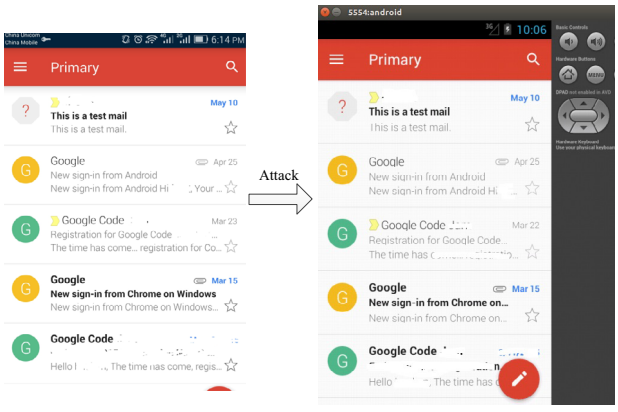

Figure 2. The attack example of Gmail.

5 Possible Solutions

5.1 Android Emulator Detection

Since the Android emulator has low financial cost and simplifies the construction of the shadow system, detecting the Android emulator can help to defend the proposed attack. Basically, the emulators have different features with the real devices. For example, the contacts, inbox text messages and call logs are usually empty in the Android emulator. In addition, the feature of WIFI, GPS, and CPU can also be used for the Android emulator detection. Currently, some researches [21, 17, 19, 15, 18] have also stressed the Android emulator detection.

5.2 Security Protocol

When an app is developed, a reasonable timeout should be designed for the user authentication. If the users are not re-authenticated for a long period of time, the security risk will increase.

To avoid the inconvenience caused by the frequent authentications, some security tokens such as random numbers can be added in the network protocols. For each login, the app server allocates different security token and update it regularly. If an app returns an old token while the security token has been updated, the server will know that the user data is hijacked and need to re-authenticate the user. Therefore, the attackers have to analyze the complicated communication protocols and deploy a Trojan in the user devices for hijacking the token, which increases the attacking cost and the exposing probability.

5.3 Trusted Execution Environment

TrustZone is a hardware-based security extension technology incorporated with the ARM processors. Based on the TrustZone, the trusted execution environment (TEE) can be constructed [7]. The technique is effective to protect the core

TABLE II. The effectiveness and disadvantages of the solutions.

Solution	Effectiveness	disadvantage
Static Attribute	Weak	Easy to be Simulated
Security Protocol	Strong	Protocol Redesign
TEE	Strong	Hardware Support
Habit-based	-	Not Accurate

services in the mobile devices and some manufacturers have added this function into their newest smart phones. The isolation mechanisms of TrustZone ensure that the guest OS components cannot access the TEE's resources even though the attackers have the elevated privilege of OS. Meanwhile, the hardware-level authentication can ensure the uniqueness of devices and avoid the malicious attacks from the shadow system effectively.

5.4 Habit-based Authentication

Habit-based authentication is to find the attackers by learning the user history behaviors. For instance, if an app is used in a rare time period such as midnight or a rare network IP, the servers should re-authenticate their users actively. Based on the machine learning approach, even the operation habits such as rhythm click characteristics [10] can be considered as the features to distinguish different users. Some of recent researches [6, 16, 20, 9] also proposed the authentication on touch gesture-based behavioral biometrics. Actually, some apps such as QQ already have the functionality to prompt users that their accounts are logged in at another city according to the network IP.

Table II presents the effectiveness analysis of these solutions. Firstly, verifying static device attributes just increase the attack difficulty, and all these attributes can be simulated simply by the attackers. By comparison, the solutions based on the security protocols and the TEE is strong against the data hijacking attack, especially for the TEE. However, the technique of TEE depends on the hardware support. Finally, the authentication based on the user history behaviors may not be accurate at present. Most of these techniques are still being researched. On other hand, compared to the traditional computers, some information of mobile devices such as locations and networks are more variable. They are inaccurate and only auxiliary to the user authentications.

Conclusion

In this paper, we propose a kind of data hijacking attack for the mobile apps. By simply hijacking the data files of the apps to the constructed shadow system, the privacies of the victims such as emails can be accessed by the attackers persistently. We verify the effectiveness of the proposed attack with 7 most popular Android apps released by the top developers, and the results indicate that most of these apps do not provide strategies against this attack or their defense can be broken simply. Finally, we give some security suggestions to the users for defeating the proposed attack.

Acknowledgment

This work was supported by National Key Research and Development Program under Grant No.2016YFB0800603, National Natural Science Foundation of China (NSFC) under Grant No.61402477, Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010703, and the National Key Research and Development Program of China under Grant No. 2016YFB0800403.

References

[1] 360 root. http://root.360.cn/.
[2] Baidu root. http://root.baidu.com/.
[3] How to root any device. http://www.xda-developers.com/root/.
[4] Jorrit chainfire jongma. how-to su: Guidelines for problem-free su usage. http://su.chainfire.eu/, 2014.
[5] Transferring data using sync adapters. http://developer.android.com/training/syncadapters/index.html.
[6] A. A. Alariki and A. A. Manaf. Investigation of touch-based user authentication features using android smartphone. In Advanced Machine Learning Technologies and Applications - Second International Conference, AMLTA 2014, Cairo, Egypt, November 28-30, 2014. Proceedings, pages 135–144, 2014.
[7] ARM. Trustzone. http://www.arm.com/zh/products/processors/technologies/trustzone/.
[8] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, and A.-R. Sadeghi. Xmandroid: A new android evolution to mitigate privilege escalation attacks. Technische Universit¨at Darmstadt, Technical Report TR-2011-04, 2011.
[9] J. G. Casanova, C. S. ´Avila, G. Bailador, and A. de Santos Sierra. Authentication in mobile devices through hand gesture recognition. Int. J. Inf. Sec., 11(2):65–83, 2012.
[10] T. Chang, C. Tsai, Y. Yang, and P. Cheng. User authentication using rhythm click characteristics for non-keyboard devices. In Proceedings of the 2011 International Conference on Asia Agriculture and Animal IPCBEE, volume 13, pages 167–171, 2011.
[11] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst., 32(2):5:1–5:29, 2014.
[12] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst., 32(2):5:1–5:29, 2014.
[13] M. C. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic detection of capability leaks in stock android smartphones. In 19th Annual Network and Distributed System Security Symposium (NDSS'12), San Diego, California, USA, Feb. 2012.

[14] L. Li, A. Bartel, T. F. Bissyand´e, J. Klein, Y. L. Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel. Iccta: Detecting inter-component privacy leaks in android apps. In 37th IEEE/ACM International Conference on Software Engineering (ICSE'15), Florence, Italy, May. 2015, pages 280–291.

[15] F. Matenaar and P. Schulz. Detecting android sandboxes. http://www.dexlabs.org/blog/btdetect.

[16] Y. Meng, D. S. Wong, R. Schlegel, and L. Kwok. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In Information Security and Cryptology - 8th International Conference, Inscrypt 2012, Beijing, China, November 28-30, 2012, Revised Selected Papers, pages 331–350, 2012.

[17] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis. Rage against the virtual machine: hindering dynamic analysis of android malware. In Proceedings of the Seventh European Workshop on System Security, EuroSec 2014, April 13, 2014, Amsterdam, The Netherlands, pages 5:1–5:6, 2014.

[18] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis. Rage against the virtual machine: hindering dynamic analysis of android malware. In Proceedings of the Seventh European Workshop on System Security, page 5. ACM, 2014.

[19] V. Rastogi, Y. Chen, and W. Enck. Appsplayground: automatic security analysis of smartphone applications. In Third ACM Conference on Data and Application Security and Privacy, CODASPY'13, San Antonio, TX, USA, February 18-20, 2013, pages 209–220, 2013.

[20] N. Sae-Bae, K. Ahmed, K. Isbister, and N. D. Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012, pages 977–986, 2012.

[21] T. Vidas and N. Christin. Evading android runtime analysis via sandbox detection. In 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014, pages 447–458, 2014.

[22] H. Zhang, D. She, and Z. Qian. Android root and its providers: A double-edged sword. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1093–1104. ACM, 2015.

[23] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 95–109. IEEE, 2012