International Journal of Digital Crime and Forensics
Volume 12 • Issue 3 • July-September 2020

User Identity Hiding Method of Android

Yi Zhang, Institute of Information Engineering, Chinese Academy of Sciences, China

ABSTRACT

With an 86.1% global market share, Android takes the top spot smartphone operating system. Due to its open environment, Android suffers from various kinds of attacks, which cause a serious privacy leakage problem. To protect users' privacy information, this article proposes a user identity hiding method of Android. The method constructs a hidden user identity in Android based on the multi-user mechanism. In particular, by hiding information related to hidden user, the method makes it invisible to normal user. The method can quickly switch the identity of normal user and hidden user by passwords. Thus, the method can deal with privacy information under hidden user identity while processing regular information under normal user identity. Compared with traditional security methods of Android, this method significantly improves the security of android devices without arousing any suspicion. Experimental results show the effectiveness of the method that it not only achieves ideal hiding effect on user identity, but also implements quick switch without sacrificing the performance of system.

KEYWORDS

Android, Multi-User, User Identity Hiding

INTRODUCTION

Nowadays, smartphone is widely used in large variety of areas, such as communication, shopping online, map navigation, getting latest news, etc. which make it an integral part of people's lives. Thus, a great deal of privacy information in smartphones, such as payment accounts, bank statement, personal photo, location information, address book and so on, become one of our most valuable treasure. Accordingly, malicious attacks on the privacy information of our smartphone are growing fast year by year. As the most widely used intelligent mobile operating system on market, Android is the primary target of all kinds of malware attacks (Xu et al., 2016). Considering the great value of privacy information stored in our phones, it is of great significance to investigate on different measures of improving the security level of Android devices.

To protect users' privacy information, researchers adopt a method of multi-system isolation in Android devices. It provides multiple running environments on a single physical device, which are independent and do not interfere with each other. By providing isolated environment for secret information, it can protect users' privacy effectively. In recent study (Huber et al., 2015), the authors propose a security architecture for Android devices based on operating system-level virtualization technology. Nevertheless, the security architecture requires deep customization for Android system, making it very complicated to implement. The "Platinum" phones of Coolpad use hardware isolation

DOI: 10.4018/IJDCF.2020070102

technique to achieve dual-system through two different ROMs (Techweb, 2017). However, the technique requires hardware customization, which is not fit for common models. In general, the traditional multi-system methods mainly focus on providing an independent and secure operating environment for users' privacy information, but it also tells attackers that users' privacy information is stored in the security system. In fact, there are very few ways to cover up the running environment of multi-system. So, it is easy to arouse suspicion when multi-system technology is used as privacy protection method. Besides, the switch of multi-system is usually complicated which may seriously impact users' experience.

In addition, smartphone manufactures design a variety of confidential cabinets and private spaces to store users' privacy information. For example, the mobile phone of Huawei provides a confidential cabinets function. Users can use it to store secret files like pictures, audio, video and so on. Only users who know the password of confidential cabinets will be able to see the secret files, but it tells attackers the existence and exact position of secret information. Once attackers crack the password of the confidential cabinets, he can directly obtain users' secret information.

In this paper, we propose a user identity hiding method based on Android device to protect users' privacy information. The method implements dual-system on mobile phone based on the multi-user mechanism of Android system. Compared with the multi-system of mobile phone with virtualization technology, our method is simple to implement and has little impact on system performance. Compared with hardware isolation technology of the "Platinum" phones, our method is implemented on software level. So it is independent to specific hardware devices, which makes it more flexible and applicable to different makers of smartphones. Furthermore, our dual-system method can make the storage of the users' regular information and privacy information further isolated from each other. The method constructs a hidden user identity, which is invisible to normal user. One can deal with privacy information under hidden user identity while processing regular information under normal user identity. Even malware detection cannot find users' privacy information through conventional detection methods. In general, this paper's contributions are:

- We develop a new kind of dual-system on mobile phone based on the multi-user mechanism of Android system to protect users' privacy. Compared with the multi-system of mobile phone with virtualization technology and hardware customization, our dual-system method is more convenient and effective without sacrificing the performance of the system;
- We construct a hiding method to achieve the identity of hidden user is invisible to the normal user. The method can prevent the conventional detection approach and common logical technique from obtaining users' privacy information. Thus, the method significantly improves the security of android devices without arousing any suspicion;
- We realize the quick, secure login and switching function between dual users. In addition, we improve the security of users' privacy information through the behavior monitoring and data encryption.

The rest of the paper is organized as follows. In Section 2, we will present some related work on privacy protection measures of Android system. Section 3 shows the design of our method. In Section 4, we present the experimental results and analysis of the proposed method. Finally, the paper is concluded in Section 5.

RELATED WORK

In recent years, privacy protection of Android has become a hot issue and main concern of researchers and mobile phone manufacturers, which is drawing more and more attention from all over the world. Researchers have put forward a variety of solutions in the privacy protection problem of Android.

Some researchers use virtualization techniques to create isolated running environments. For example, AirBag (Wu et al., 2014) presents a lightweight OS-level virtualization approach. The approach assumes that there is a trusted OS kernel and the user will install untrusted apps onto phones. In order to isolate and prevent untrusted apps from infecting normal systems or stealthily leaking private information, AirBag dynamically creates an isolated virtual environment to ensure their transparent "normal" execution. They mainly focus on the preliminary analysis of Android applications before their execution in the trusted container. However, this kind of isolation will affect the functionality of untrusted apps at some extent.

Different from the purpose of AirBag, Huber et al. (2015) propose a secure architecture for OS-level virtualization on mobile devices. They achieve data confidentiality by isolating containers and restricting them to a set of minimal controlled functionality. In addition, the architecture defines specific channels for communication between components and develops a stacked Linux Security Modules (LSM) concept using SELinux and a custom LSM. However, their secure architecture is complex and has a certain impact on system performance.

In addition, some research work focuses on improving the existing security mechanism or refining the security strategy of Android system (Liang, 2015; Ongtang, 2012). Some researchers (Gibler, 2012; Nauman, 2010; Shen, 2014) propose solutions to the problem of Android permission mechanism. In order to enhance the security of Linux kernel, researchers introduce SELinux to the Android system (Bugiel, 2013; Shabtai, 2010; Smalley, 2013), providing a more rigorous access control strategy. TaintDroid (Enck et al., 2014) mainly focuses on the personal data. It aims to detect when sensitive data leaves the system via untrusted applications and to facilitate analysis of applications by phone users or external security services.

In smartphone market, the "Platinum" brand of Coolpad launches a typical dual-system mobile phone. It uses hardware isolation method and has two independent running systems, open system and security system (Techweb, 2017). Open system is normal mobile phone system, where users can perform tasks that require less security, such as playing games and watching video. The open system also has a cool function of housekeeping to achieve real-time monitoring of mobile phone system. In security system, all apps have passed the official safety certification. Users can perform operations that require relatively high confidentiality. However, the device uses two different ROMs to implement dual-system, requiring hardware customization. So it is improper for common models and difficult to popularize.

Android system supports multi-user mechanism on a single device since version 4.2. The mechanism tells user accounts and application data apart (AOSP, 2017). The first user created during the initial process is primary user, which is always running on the device. No one can remove the primary user except the operation of factory reset. Any user added to the device later is secondary user. In particular, primary user can remove secondary users while secondary users cannot affect other users on device. Temporary secondary user is named as guest user that is optional to be deleted when it is useless. A single device only allows one guest user at a time. When multiple users exist on device simultaneously, each user is allocated with a unique workspace to place its installed applications. According to the multi-user mechanism of Android system, there is no authority for secondary user to access the privacy information or call the applications of other ones. But for primary user, it has the authority to remove applications or even the entire workspace established by secondary users.

DESIGN

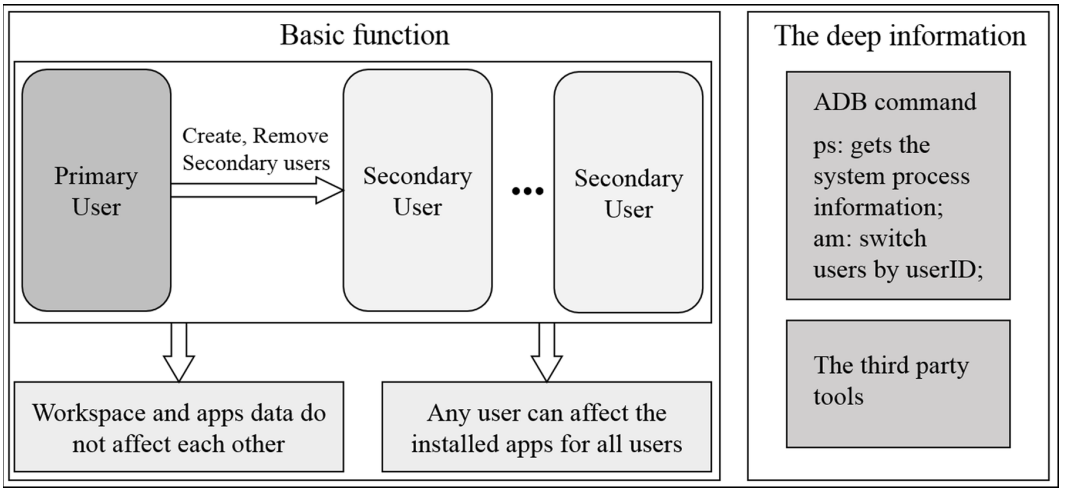
To protect users' privacy information, we propose a user identity hiding method on the basis of multi-user mechanism of Android system. In the method, we construct two default users named normal user and hidden user which exist on a single device simultaneously. Under the identity of normal user, people can carry out ordinary operation with low sensitivity. While under the identity of hidden user, people can deal with information that ranked with higher security demands, such as individual

privacy or payment accounts. Thus, our method can separate the sensitive operation from ordinary operation, which significantly reduces the risk of information leakage. In particular, we achieve the functionality that hidden user is invisible to normal user, which ensure the security that detectors cannot find hidden user by conventional detection approaches. What's more, we prevent the normal user from perceiving, manipulating and getting data of the hidden user. As a result, our method can realize the protection of privacy information by identity of hidden user. In addition, we improve the security of privacy information in the identity of hidden user through the Behavior Monitoring (BM) and Data Encryption (DE).

Hiding Method

The basic design of our method is to construct two users according to the multi-user mechanism. In our method, we inject the code of adding new user during the process of starting user management service of the system. When a user launches the system, user management service will examine whether a hidden user exists. If the hidden user is non-existent, user management service will call the function of adding new user. Therefore, there are two default users in the system: the primary user as the normal user, a secondary user as the hidden user. In order to implement the hiding effect of secondary user, we modify the specific function related to multi-user in Android system. As Figure 1 shows, the information related to multi-user in the Android system includes two parts: the basic functions provided by the Android system and deep information.

Figure 1. Operation related to multi-user of Android system



Multiple users in Android have isolated workspace and apps data. However, the primary user can create, delete and switch to the secondary user. Any user can affect the installed apps for all users. The Android system provides these functions to users through some operating interface, such as the list of users, the list of applications installed in the phone, the storage allocation and so on. The information mainly exists in the APP of system setting. In order to prevent normal user from controlling and perceiving hidden user through these functions provided by Android system, we modify the permissions of normal user and secondary user. Such as the function (getUsers()) of getting users' information at the user management service class (UserManagerService.java) and the function (getInstalledApplications()) of getting applications that have been installed to the system at the package management service class (PackageManagerService.java). The user management service

(UserManagerService.java) is responsible for managing multiple users of Android system, such as creation, deletion, query and so on. The package management service is responsible for managing packages of the Android system, including installation, uninstallation, query and so on. Under the identity of primary user, the function (getInstalledApplications()) can get applications of all users installed in system. The storage allocation information is displayed according to the function's (getUsers()) execution result. When apps call these functions, system will determine whether the current user is hidden user. If the current user is not hidden user, these functions will not obtain the information of hidden user. Even if other users and hidden user install the same application, other users' modification to the application will not affect the hidden user.

The deep information refers to the relevant content of the hidden user that users can obtain indirectly. The content includes system processes information, file directory information, configuration file information etc. The methods of getting deep information mainly include USB debugging and the third party detection tools. The access to the information is achieved mainly through adb command or service interface provided by the system. ADB (Android Debug Bridge) is a powerful command-line tool that lets users communicate with Android device. It provides some command (e.g.: ps, ls, am, pm) that can gets the information of system. For example, ps command can get the process information that system is running. Activity management service (AMS, ActivityManagerService.java) is a kernel service of Android system. It is responsible for the management and dispatching of application process and the four components (Activity, Service, Broadcast and Content Provider) in Android system. For example, AMS provides an interface (getRunningAPPProcess()), by which we can also get the process information that system is running. Therefore, we modify the relevant adb command and service interface. Users cannot obtain the information of hidden user through adb command or third-party tools. At the same time, we also disguise the relevant configuration files and ensure that users with local root privileges cannot get the information of hidden user as well. Furthermore, we close the USB debugging function by modifying the configuration file of Android system directly.

Behavior Monitoring (BM)

The goal of BM is to achieve real-time monitoring on users' privacy information. It can detect abnormal behaviors and responses to these behaviors quickly. In particular, BM prevents other users from any operations related to hidden user.

The architecture of Android system is constituted of three layers, Linux kernel, system architecture layer and applications layer (Tam et al., 2017). Based on applications layer, users can monitor behavior of applications through the broadcast mechanism of Android system (Enck et al., 2009). However, the system broadcast limits the monitoring scope, making it is not comprehensive enough. At the system architecture layer, users can hook the Dalvik/ART virtual machine to monitor behaviors of applications (Jingya et al., 2016). However, this method only applies to the services which implemented by Java, and it does not work on the C/C++ services.

By intercepting system calls at the kernel level (Jeong, 2014; Pan, 2014), users can monitor all of the system's file access operations, and get more comprehensive behavior information. At the same time, the system modification and influence on performance of monitoring at kernel layer are far less costly than that at a framework layer. Therefore, we monitor and handle the application behaviors by intercepting system calls. First, we need to get the system call table (sys_call_table), and then modify the address of system calls that we want to intercept, making it point to our custom function. Therefore, we can implement the monitoring function for specific system calls. The main goal of our method is to protect privacy information of users, so we mainly modify the system calls associated with the file operation.

Data Encryption (DE)

The function of DE is to encrypt specific files in hidden user. We set a special folder in hidden user. When users store a file within the folder, the system will prompt the user to input a password. The

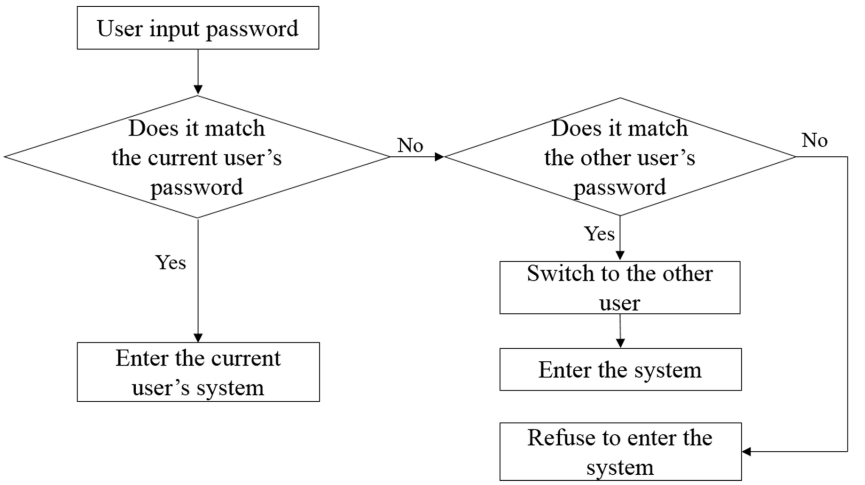
system will encrypt all documents stored in this folder automatically. When applications access encrypted files, the system prompts for password. If the input is error for three times consecutively, the system locks the files for a predefined time during which any access is prohibited. If the password is correct, the system decrypts the file and save the plaintext file temporarily. After a certain time, system will delete the plaintext file.

We use the symmetric encryption algorithm AES-256 to encrypt files. The security of the symmetric encryption algorithm relies heavily on the safety of the key (Mane, 2015). Therefore, the system does not directly keep the key and only saves its hash value. We use the SHA-256 algorithm to calculate hash value.

User Switch

In order to achieve login and switch function between dual users, we modify the execution logic of Android system password authentication, making sure that user can enter different system by inputting corresponding password on login interface. Figure 2 shows the process of our design. When the user

Figure 2. Login process



inputs a password on login interface, system will execute the function (checkPassword()), which provides a password authentication service. If the password matches the current user unsuccessfully, the function that we have modified will match with the other user. If the password matches successfully, the function will call the switch function (switchUser()), which provides switch service between different users. Meanwhile, in order to achieve seamless switch between dual users, we modify the user switch function. When user switches to another one, system will pop up a window to remind the user. We remove this window by modifying the program and avoid re-authentication after switching. The switch function by password only works for normal user and hidden user.

RESULT EVALUATION

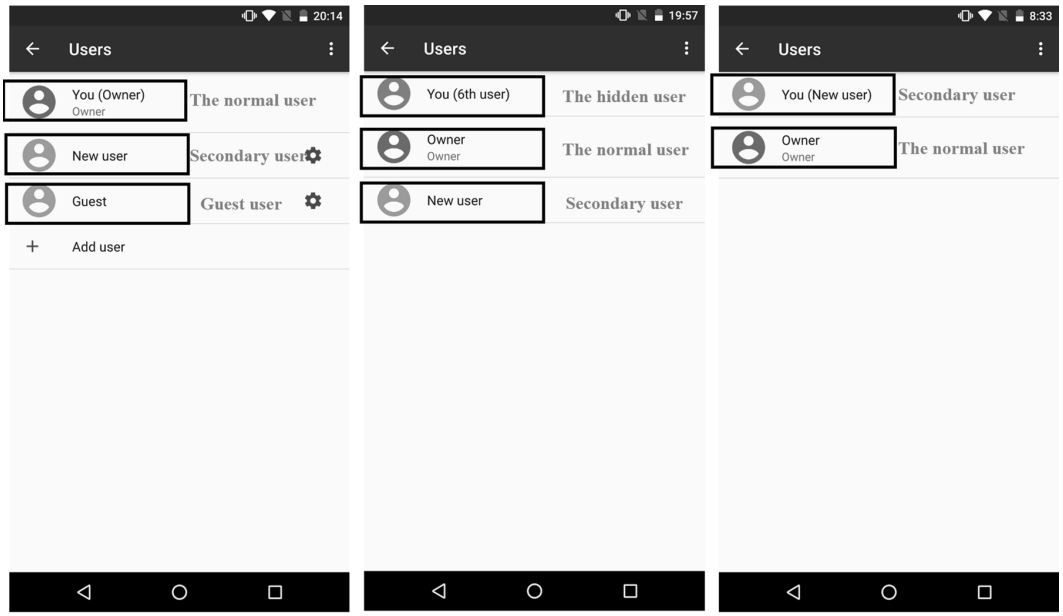
We implement a prototype of the proposed method based on Android 6.0.1 source code and test it on Nexus 6P device. There are two default users after the initialization of the device. The user can enter the hidden user only after he has set up the normal user's lock password. We set up a default password for hidden user during the initialization of the normal user's lock password. Then on login

interface, when user input the default password of hidden user, system will switch to hidden user after it confirms the password, and vice versa. In this way, the user can enter different system with different passwords. Users will not see the switching trace during the switch process between dual users. It is important to note that only users who know the default password can enter the hidden user. According to our program, users can only switch identity between normal user and hidden user through passwords. In addition, other original functions of the system remain unchanged. We also test WeChat, music, video and other apps, all of which can run successfully as normal.

Security

The goal of our method is to hide the identity and privacy information of hidden user. When users enter the normal user or other secondary users, they will not find the hidden user. Figure 3(a) is users list of the normal user (the primary user). As we can see, the normal user creates a secondary user called new user and the hidden user is not in the users list. As Figure 3(c) shows, users list of the

Figure 3. Users list

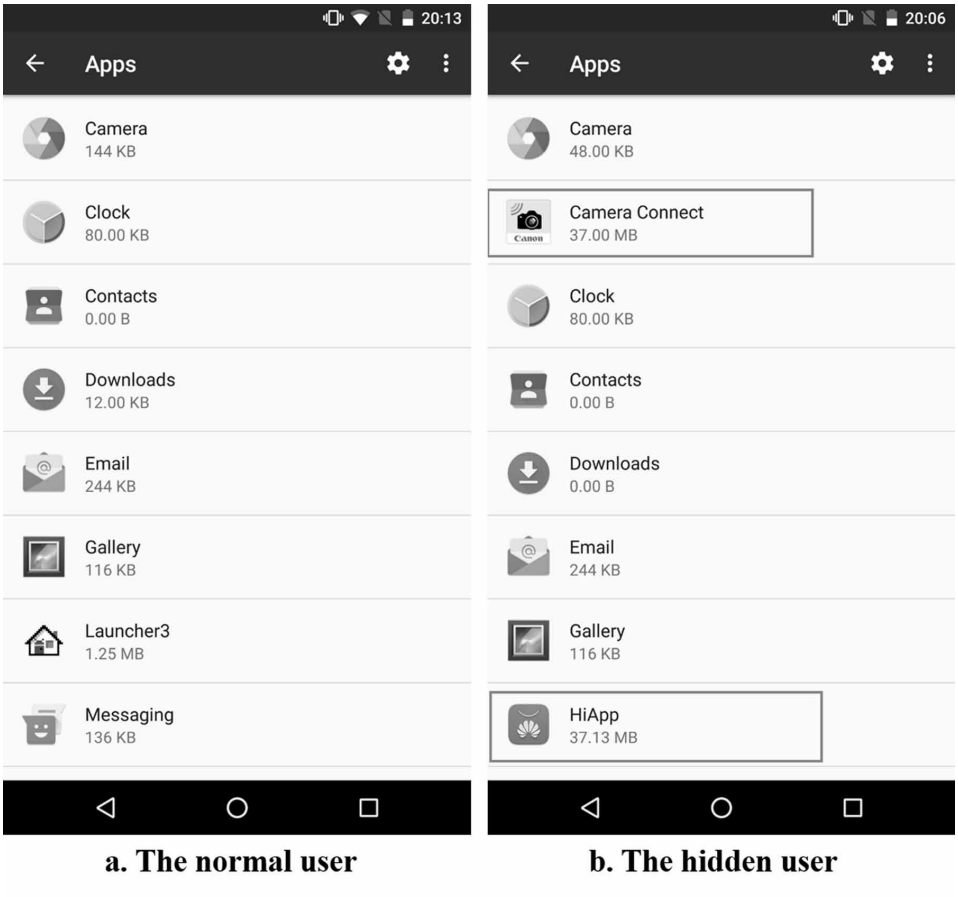


a. The normal user **b. The hidden user** **c. A secondary user**

secondary user also does not have the hidden user. Therefore, the user cannot switch to the hidden user. The normal user also cannot control the hidden user. Similarly, other place of the device also will not have the information of hidden user. Figure 4(a) shows the apps list of the normal user and (b) is apps list of the hidden user. The hidden user installed two new apps, which will not appear in the normal user and the secondary user. In addition, users also cannot find the files of the hidden user in the file system of the normal user and other users. Therefore, users cannot find and control the hidden user through functions provided by Android system if they do not know the password of the hidden user. To be clear, we set a complex password for the hidden user that consists of multiple symbols.

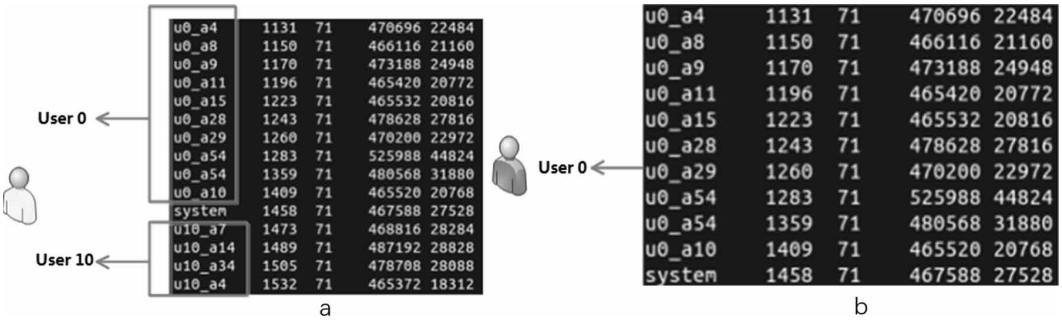
We hide the identity and privacy information of the hidden user in the normal user and other users. At the same time, users also cannot find the hidden user identity through adb command. In addition, the BM prevents other users from any form of operations related to hidden user. As Figure

Figure 4. Apps list



5(a) shows, in the common Android devices, users can see the process information of all users in the system by "ps" command or other tools. In Our system, users cannot see the hidden user's information, as the Figure 5(b) shows. Users also cannot switch to the hidden user by "am" command and get information of the hidden user by "pm" command. Even if the detectors have root privileges, the "ls"

Figure 5. Result of ps command



command will not display the catalog information of the hidden user and the "adb pull" command will not obtain the data of the hidden user.

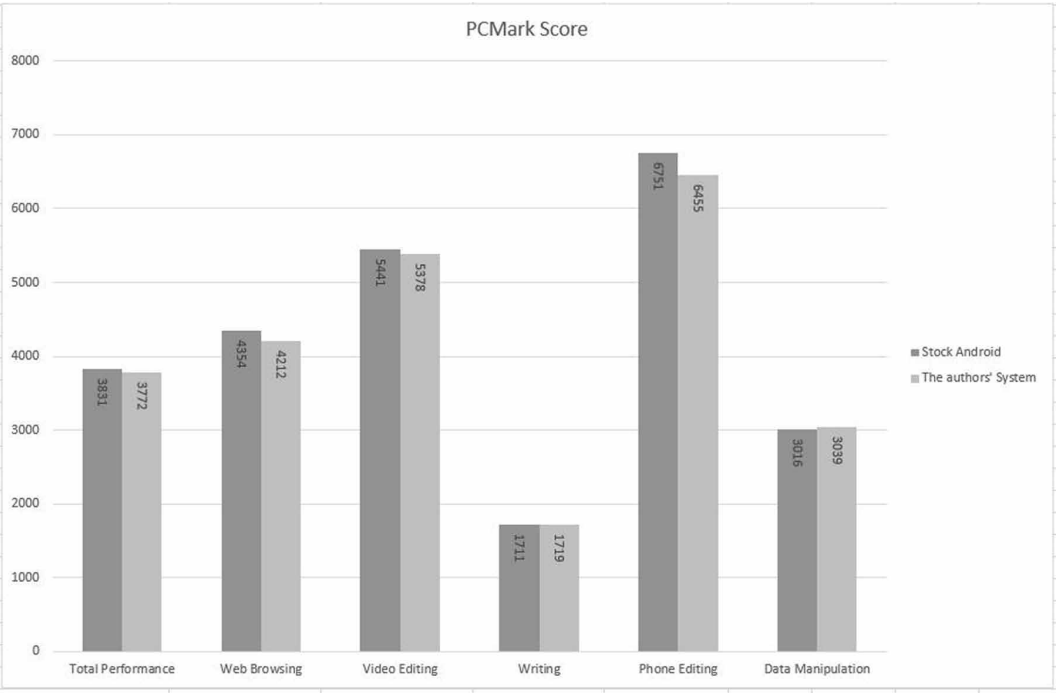
Most third-party tools on Android devices rely on functionality provided by Android system. For example, in Android forensics, the most common logical technique relies on the Content Providers built into the Android platform. AFLogical is an Android forensics logical technique. The AFLogical app takes advantage of the Content Provider architecture to gain access to data stored on the device. Similar to Commercial Android logical tools, the device must enable USB debugging for AFLogical to extract the data. The Content Providers currently supported by AFLogical include Brower Bookmarks, Brower Searches, Calendars, Calendar Events and Call Log Calls and so on (Hoog, 2011). When the user installs AFLogical to the normal user, the app can get the above information of the normal user only after he enables USB debugging successfully. However, due to our modifications to the Android system, the app will not find identity and privacy information of the hidden user without arousing any suspicion. Because we limit the root privileges, other tools with root privileges also cannot obtain the data of the hidden user. Our method can prevent the common logical techniques and avoid the leakage of data stored in the hidden user.

Performance

PCMark for Android introduces a fresh approach to benchmarking smartphones. It measures the performance and battery life of the devices as a complete unit rather than a set of isolated components (Futuremark, 2017). We run the PCMark on the Nexus 6P with stock Android and with our system on Android 6.0.1. As Figure 6 shows, the test result of performance is 3831 with stock Android and 3772 with our system on the Nexus 6P. The performance impact of our system in PCMark is about 1.5%.

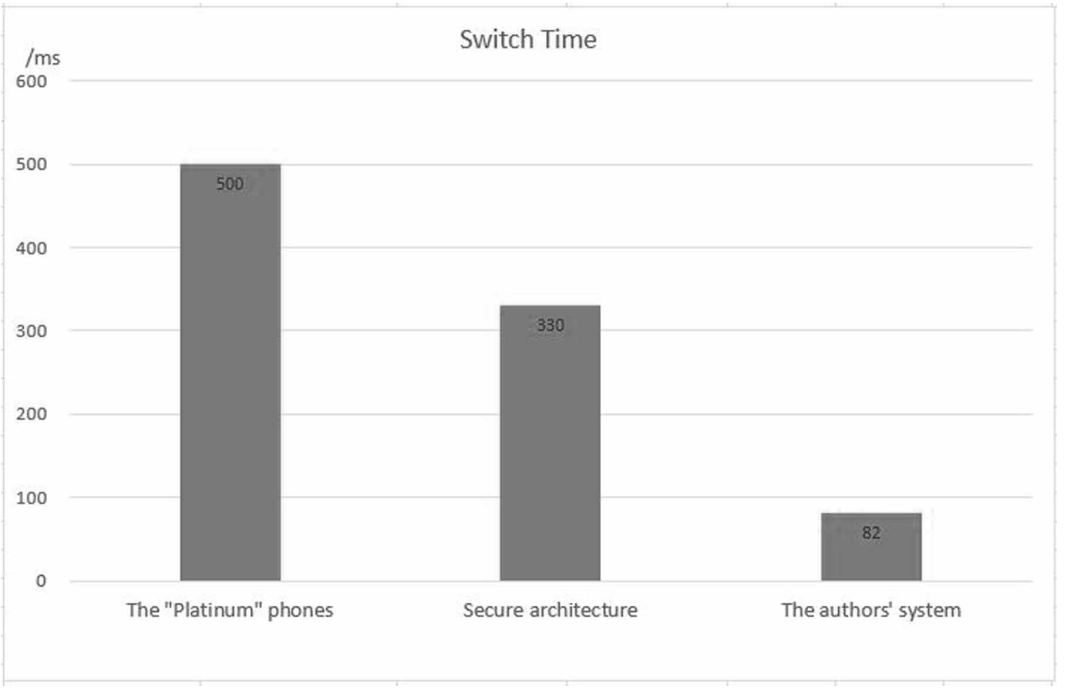
We measure the switching time between the normal user and the hidden user. The average of the tests is 82 ms. However, it consumes about 330 ms switch from privileged container to unprivileged container and 300 ms switch from unprivileged container to privileged container, in Huber et al.

Figure 6. Result of PCMark workbench test



(2015) devise the secure architecture. The "Platinum" phones of Coolpad takes about 500 ms to switch between different systems, using fingerprint, as the Figure 7 shows. It will take more time using normal switching mode.

Figure 7. Switch time



CONCLUSION

In this paper, we develop a user identity hiding method based on Android device to protect users' privacy. Based on the multi-user mechanism of Android system, we construct a hidden user identity in Android devices. At the same time, by modifying the system logic of locking screen, we can quickly switch the identity of normal user and hidden user by entering two different passwords on login interface. Covered with a normal user identity, detectors cannot find the existence of hidden user identity by conventional detection methods. Then, the method can hide the information stored in the hidden user which help to protect users' privacy. In order to fulfill the goal, we hide the specific information related to multi-user mechanism of Android system. We further propose the behavior monitoring and data encryption methods to protect users' privacy. By intercepting system calls related to file operations, we achieve system monitoring on file operations. Moreover, the security of users' privacy information is further improved by utilizing data encryption in our method. Experimental results show the effectiveness and convenience of our method that it not only achieves ideal hiding effect on user identity, but also implements quick switch without sacrificing the performance of system.

REFERENCES

A better benchmark for Android devices. (2017). Futuremark. Retrieved from https://www.futuremark.com/benchmarks/pcmark-android

Bugiel, S., Heuser, S., & Sadeghi, A. R. (2013, August). Flexible and Fine-grained Mandatory Access Control on Android for Diverse Security and Privacy Policies. *Proceedings of the USENIX Security Symposium* (pp. 131-146). Academic Press.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., & Sheth, A. N. et al. (2014). TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems*, *32*(2), 5. doi:10.1145/2619091

Enck, W., Ongtang, M., & McDaniel, P. (2009). Understanding android security. *IEEE Security and Privacy*, *7*(1), 50–57. doi:10.1109/MSP.2009.26

Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017. (2017, May 23). Gartner. Retrieved from http://www.gartner.com/newsroom/id/3725117

Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012). AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. *Trust*, *12*, 291–307.

Hoog, A. (2011). *Android forensics: investigation, analysis and mobile security for Google Android*. Elsevier. doi:10.1016/B978-1-59749-651-3.10001-9

Huber, M., Horsch, J., Velten, M., Weiss, M., & Wessel, S. (2015, November). A secure architecture for operating system-level virtualization on mobile devices. *Proceedings of the International Conference on Information Security and Cryptology* (pp. 430-450). Springer International Publishing.

Jeong, Y. S., Lee, H. T., Cho, S. J., Han, S., & Park, M. (2014, March). A kernel-based monitoring approach for analyzing malicious behavior on android. *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (pp. 1737-1738). ACM. doi:10.1145/2554850.2559915

Jingya, Y., Senlin, L., & Shuai, Z. (2016). Research on Method of Android System Malware Behavior Monitoring Based on Multi-level and Cross-view Analysis. *Netinfo Security*, (7), 40-46.

Liang, H., Wu, D., Xu, J., & Ma, H. (2015, November). Survey on privacy protection of android devices. *Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)* (pp. 241-246). IEEE. doi:10.1109/CSCloud.2015.21

Mane, R. R. (2015). A Review on Cryptography Algorithms, Attacks and Encryption Tools. *International Journal of Innovative Research in Computer and Communication Engineering*, *3*(9), 8509–8514.

Nauman, M., Khan, S., & Zhang, X. (2010, April). Apex: extending android permission model and enforcement with user-defined runtime constraints. *Proceedings of the 5th ACM symposium on information, computer and communications security* (pp. 328-332). ACM. doi:10.1145/1755688.1755732

Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P. (2012). Semantically rich application-centric security in Android. *Security and Communication Networks*, *5*(6), 658–673. doi:10.1002/sec.360

Pan, X., Zhongyang, Y., Xin, Z., Mao, B., & Huang, H. (2014, September). Defensor: Lightweight and efficient security-enhanced framework for Android. *Proceedings of the 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 260-267). IEEE.

Shabtai, A., Fledel, Y., & Elovici, Y. (2010). Securing Android-powered mobile devices using SELinux. *IEEE Security and Privacy*, *8*(3), 36–44. doi:10.1109/MSP.2009.144

Shen, F., Vishnubhotla, N., Todarka, C., Arora, M., Dhandapani, B., Lehner, E. J., & Ziarek, L. et al. (2014, September). Information flows as a permission mechanism. *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering* (pp. 515-526). ACM.

Smalley, S., & Craig, R. (2013, February). Security Enhanced (SE) Android: Bringing Flexible MAC to Android. In NDSS (Vol. 310, pp. 20-38).

Supporting Multiple Users. (2017, June). AOSP. Retrieved from https://source.android.com/devices/tech/admin/multi-user

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The evolution of android malware and android analysis techniques. *ACM Computing Surveys*, *49*(4), 76. doi:10.1145/3017427

The First Mobile Security Forum Coolpad Helps Secure Market. (2016, April 25). Techweb. Retrieved from http://www.techweb.com.cn/digitallife/2016-04-25/2321952.shtml

Wu, C., Zhou, Y., Patel, K., Liang, Z., & Jiang, X. (2014, February). AirBag: Boosting Smartphone Resistance to Malware Infection. In NDSS. Academic Press.

Xu, M., Song, C., Ji, Y., Shih, M. W., Lu, K., Zheng, C., & Lee, S. et al. (2016). Toward engineering a secure android ecosystem: A survey of existing techniques. *ACM Computing Surveys*, *49*(2), 38. doi:10.1145/2963145

Yi Zhang is a master of computer technology in University of Chinese Academy of Sciences. She received her bachelor of information security in 2015 from Beijing Jiaotong University.