Security Analysis of Cryptocurrency Wallets in Android-based Applications

Daojing He, Shihao Li, Cong Li, Sencun Zhu, Sammy Chan, Weidong Min, and Nadra Guizani

Abstract

As digital currency wallets are gaining popularity, security incidents related to them are also increasing. They lead to the disclosure of users' private information and security threats of users' assets. This article analyzes the security risks of digital wallets in Android, which is the most popular mobile operating system. It first establishes the threat model and the security goals, and then analyzes the attack surface and attack vectors from the aspects of wallet apps and the Android system. Experiments are conducted on two real-world digital currency wallet apps, and the results reveal several serious security risks in these apps, which highlight the necessity and importance of developing secure cryptocurrency wallets.

Introduction

Blockchain-based digital currencies are built upon the concept of blockchain [1], which is an open and distributed ledger with records that can be verified but cannot be modified. Since the birth of bitcoin, the number of cryptocurrencies has grown into several thousands with a market capitalization of more than $233 billion as of November 2019 [2].

In bitcoin and many other digital currency systems, users use the account private key as the sole basis to control the digital currency account, so any operation signed with the private key is regarded as the legal operation of the account. However, there is a lack of support for loss reporting, key updating, and mechanisms which are commonly used in other payment systems, such as abnormal transactions identification and two-factor authentication [3], to prevent risky transactions. As a result, obtaining the private key of an account is equivalent to having total control of the account.

Private keys can be hosted on exchanges or user-controlled cryptocurrency wallets. The former gives exchanges full control over their digital currency accounts. Unfortunately, a number of exchanges have been broken-in by hackers, or suffered from inside attacks or system failures. The loss of assets caused by such events makes many users reluctant to put a large amount of assets in the exchanges [4]. Differently, digital currency wallets do not store digital currencies, but are only responsible for generating, storing and managing the private keys of digital currency accounts. The encrypted private key of a user is stored in its wallet. When the user needs to make a transaction, it is decrypted to sign the transaction record. Unless the user exports it manually, the private key is always kept in the wallet. This provides greater security for the private key and the account assets.

However, the security of digital currency wallets cannot be neglected. Due to the vulnerabilities in both wallets and operating environment, as well as the lack of security awareness of users, the theft of account keys and the tampering of trading parameters have occurred [5]. Indeed, researchers have made relevant explorations on this issue. In the field of digital wallet security, Bui *et al.* [6] studied the desktop cryptocurrency wallet remote procedure call (RPC) interface and found that attackers could effectively steal privacy information regardless of permissions. Androulaki *et al.* [7] analyzed and demonstrated security threats to wallets. On security vulnerability analysis of the Android operating system, Sarma *et al.* [8] described its security risks due to permission requesting. Fratantonio *et al.* [9] used two permissions provided by Android to fully control the user interface feedback loop to create devastating attacks and launched 12 attack experiments successfully. Ren *et al.* [10] provided mitigation suggestions toward a more secure Android multitasking sub-system. Xu *et al.* [11] described the security risks of Android USB debugging, and further presented methods to improve the security. Mayrhofer *et al.* [12] discussed the difficulty of protecting mobile device platforms against different threats and suggested potential solutions. Xu *et al.* [13] inferred user's touchscreen tapping input through the permissionless motion sensors on a smartphone.

To provide a deeper understanding of the threats faced by cryptocurrency wallets on the market today, this article studies the security weaknesses of both the Android system and specific digital currency wallets. Targeting at several popular cryptocurrency wallet apps, we implement attacks to acquire the private keys from wallets, or change the transactional information of a digital currency transfer. Our work shows the current security status of digital wallet apps, and also calls for effective protection solutions. Our contributions are as follows:

- We identify the vulnerabilities of cryptocurrency wallets from multiple perspectives.
- We analyze the security issues of popular cryptocurrency wallets, and establish an adversary model for cryptocurrency wallet.
- Based on the vulnerability analysis, we design attack experiments to demonstrate the validity of our analysis.

Daojing He is with Nanchang University and East China Normal University; Shihao Li and Cong Li are with East China Normal University; Sencun Zhu is with Pennsylvania State University; Sammy Chan is with City University of Hong Kong; Weidong Min is with Nanchang University;; Nadra Guizani is with Purdue University.

Digital Object Identifier:
10.1109/MNET.011.2000025

0890-8044/20/$25.00 © 2020 IEEE

The article is organized as follows. The following section describes the concept of cryptocurrency wallets, defines an adversary model and security target. We then analyze the vulnerabilities of cryptocurrency wallets. Following that we present our attack experiments on two real-world Android based cryptocurrency wallets. The final section concludes the article and suggests some future work.

Overview of Cryptocurrency Wallet Security

Cryptocurrency wallet is a piece of software or hardware that can generate, store and manage private keys of the cryptocurrency accounts. For the software wallets, such as the wallets on mobile systems, they also allow users to query the transaction history and the account balance, generate, sign, and send new transactions to the distributed network.

Because owning the private key is equivalent to having full control of the corresponding cryptocurrency account, private key management becomes very essential for security. Private key needs to be encrypted before being stored in the wallet and decrypted into plaintext when used. Taking Ethereum as an example, the plaintext of the private key is a 256-bit binary number, which is usually encoded as a hexadecimal number for presentation. It is encrypted and stored in a keystore, and imported or exported through the keystore. The relationship among private key, keystore and mnemonic is shown in Fig. 1.

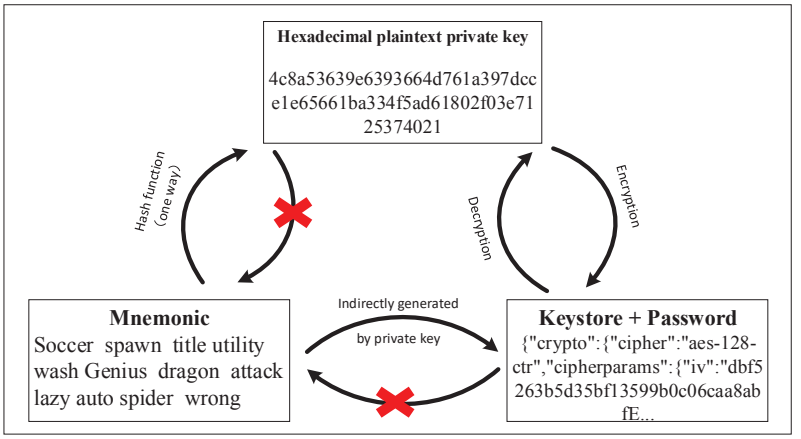

FIGURE 1. Relationship among private key, keystore and mnemonic.

Security Goals of Cryptocurrency Wallets

The security goals of the cryptocurrency wallets are consistent with most security systems, mainly including confidentiality, integrity, and availability.

Confidentiality: Confidentiality is to prevent unauthorized access to confidential information. For a cryptocurrency account, its private key means the full control of the account and all the digital assets under the account. Therefore, the fundamental security property of the wallet is to ensure that the private key is not accessed in an unauthorized way. Transaction information is not considered confidential due to the fact that all data on the blockchain is publicly accessible.

Integrity: Integrity is to prevent information from being modified by unauthorized entities so as to ensure the accuracy and completeness of the information. As to the cryptocurrency wallets, it is of great importance to guarantee the integrity of the private key. If the private key stored in the wallet is illegally modified or deleted, the user will lose control of the account, thereby losing the assets in the account. For transaction data that have been sent to the blockchian, blockchain has used cryptographic methods such as signatures and hashes to ensure transaction data has not been tampered with. However, the integrity property is crucial for a newly initiated transaction. If the information of the transaction has been tampered with before the user signs the transaction with the private key, the transaction will be recognized by the blockchain system because it has the legitimate owner's signature. For historical transactions, they can also be tampered with after they are obtained from the blockchain system. Especially for some wallets, such as mobile wallet apps, due to their own computing and storage resource limits, they cannot directly interact with the blockchain. Instead, they communicate with an agent server to obtain the historical transaction information. In this case, integrity protection during the transfer from the agent server to the wallets is also required.

Availability: Availability is to ensure that the legitimate use of information is not denied, meaning the information should be accessible and usable upon demand by an authorized entity. For wallet apps, it is necessary to ensure that keys can be generated, stored, and accessed correctly. Also, the transactions should be signed, sent and queried correctly upon users' requests.

Adversary Model for Cryptocurrency Wallets

For different types of digital currency wallets, their adversary models are slightly different. We discuss the adversary model for software-based cryptocurrency wallets in this section.

The adversary's goal is to break the confidentiality, integrity, or availability properties of the data in the wallets. This includes obtaining the private key, tampering with newly initiated transactions, tampering with historical transactions, blocking the creation of new transactions, denying transaction information inquiries, and so on.

The adversary does not have the private knowledge unique to the owner of a target wallet, such as the private key of the user's account or the set of wallet transaction passwords. However, the adversary has the ability to install and run any application on the same device that the wallet runs. The installed application has been given all necessary permissions it requests. The adversary has the ability to modify any setting on the device on which the wallet runs. The adversary also has the ability to run any software on the other devices of the user who uses the wallet. The adversary can listen to and modify the communication of the wallet, although they do not know the corresponding key for the encrypted traffic. The adversary can attack the servers to which the wallet is connected, but cannot control the blockchain network.

The above adversary model is practical, because the user may be induced to install a new application and further grant the application all its required permissions. The application can disguise itself as a normal application. In addition, techniques such as USB debugging, accessibility service and other mobile phone features, can offer more opportunities exploited by adversaries. Indeed, we will describe these attack techniques in the next section.

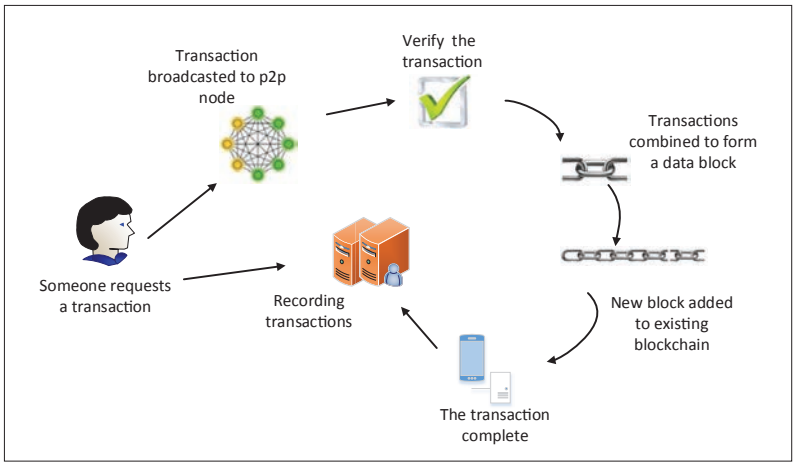

FIGURE 2. The transaction process.

Cryptocurrency Wallets Vulnerabilities

Cryptocurrency wallets generally carry the basic functions of private key management and transaction management. Key management includes generating, storing, importing and exporting a private key, and transaction management includes transferring and collecting tokens, and querying transactions and balances. However, improper implementation of these functions may introduce attack points into the attack surface. Besides, as the wallet is hosted in an operating system (OS), the features provided by the OS may also be exploited by the attacker, thus posing a threat to the security of the digital wallet. In the following, we analyze the attack surface from the aspects of the cryptocurrency wallet itself and its underlying OS.

Attack Surface of Cryptocurrency Wallets

Key Management: If the user does not have an account for an encrypted digital currency, a pair of public and private keys for a new account will be randomly generated locally on the device through the wallet. If the user already has an account, the private key corresponding to the account can be imported into the wallet so that the user can manage the account using this wallet. The wallet encrypts the generated or imported private key with an encryption password chosen by the user. If a user loses the private key, they permanently lose control of the corresponding account, resulting in asset loss. Therefore, the private key is usually exported for backup.

In the process of generating a private key, if the random seed used in the generation can be predicted or obtained, the generated private key is also at risk of being compromised. In the process of storing the private key, if the stored key can be accessed in plaintext or can be decrypted, it may be stolen and compromised, thus posing a threat to confidentiality. If the stored key can be modified or deleted by a third party, it may cause the user to lose control of the account, which poses a threat to the integrity and availability of the account. When importing a private key, a user may need to manually enter the key using the keyboard, or by copying and pasting, an attacker may obtain the key by monitoring the user input, posing a threat to confidentiality. When importing and exporting keys, the wallet may also display key-related information on the screen. An attacker may monitor the information on screen to obtain the key, posing a threat to confidentiality. Last but not least, when the key encryption password is set or verified, an attacker may capture the key encryption password by listening to user input, again a threat to confidentiality.

Transaction Management: When a user needs to transfer funds out of the account, the wallet generates the corresponding transaction and signs it with the user's private key. It then broadcasts the signed transaction to the blockchain network, and waits for confirmation to complete the transfer function. When a user needs to make a collection operation, it needs to provide its account address, which optionally includes the currency and amount, to the payer for payment. The wallet app generally also provides transaction records and balance query functions for users to query the corresponding account transfer records and account balances. The process is shown in Fig. 2. Since this information cannot be queried directly in some blockchain systems, this implementation may require a connection to a wallet server dedicated to the service, rather than a blockchain system.

When transferring money and collecting money, the transaction information entered by the user or displayed by the wallet may be tampered with, which poses a threat to integrity and may result in the user's funds being transferred to the attacker's account. When transferring money, if the keyboard and user password input screen are monitored, the encrypted password may be stolen, posing a threat to confidentiality. Diao *et al.* [14] used the interrupt processing mechanism to deduce the user's unlock pattern and the state of the foreground application without any authority, exposing the severity of security issues in the transfer process. When transferring money and querying transactions or balances, if an attacker can interrupt the transfer or query by cutting off the connection between the wallet and the blockchain network or the wallet server, it will pose a threat to availability, and may potentially lead to sensitive operations such as the export of the private key by the user in order to regain control of the account, resulting in further damage. When querying transactions and balances, an attacker may mislead the user by tampering with information stored on the wallet server, information transmitted between the server and the wallet client, or the information displayed on the wallet, the integrity will be threatened, resulting in the wrong transaction records or wrong balance displayed on the wallet, thus misleading the user.

Attack Vectors of an Android System

Android offers a number of features for developers and users, but some of these features can be exploited by attackers to compromise the security of the cryptocurrency system running on Android. We analyze their potential attack vectors after introducing these features.

Root Privilege: The Android mobile operating system uses the Linux operating system kernel [15]. The root user is the most privileged administrator user built into Linux, as is Android. If an app gets the root privilege, it will run as the root user with full control over the device. Root privilege is usually limited to some core processes of

Attack vector and attack surface	Private key generation	Private key import	Private key storage	Private key export	Token transfer	Token collection	Transaction query
Root	C	C	CIA	C	CIA	I	IA
USB debugging	C	C	CIA	C	CI	I	I
Accessibility service		C		C	CI	I	I
Third-party input method		C		C	C		

TABLE 1. Relationship of Attack Surface and Attack Vectors.

the system, but some device manufacturers allow users to turn on the device's root permissions and authorize user-specified apps, thereby enhancing the user's control over the device. It is also possible for a user or a malicious program to obtain root privilege through a vulnerability.

USB Debugging: USB debugging is a debugging function that Android OS provides to developers to connect Android devices to computers through USB cables. When an Android device turns on the USB debugging option and connects to a computer, the computer can issue ADB (Android Debug Bridge) commands to debug the device. ADB is a command-line tool that allows users to perform many privileged operations over Android devices by running the commands on their computers, such as installing and unloading backup apps, reading and writing files, modifying system settings, taking screenshots, simulating user actions, executing shell commands, and so on.

Because USB debugging is powerful, improper use may easily cause security risks. USB debugging is turned off by default and requires the user to turn it on manually in the developer option. If the Android version is 4.2 and above, the developer option is hidden by default, requiring the user to click "version number" 7 times in a row in "About the phone" to reveal the debugging option. If the Android version of the device is above version 4.2.2, when the device is connected to USB debugging, the device screen will prompt a dialog about whether to allow the computer to debug. Users can permanently disable the prompts.

Accessibility Service: Android's accessibility service provides user interface enhancements to people with disabilities or who are temporarily unable to fully operate their devices. This function can be used to acquire content on the screen, listen to the user's actions, interact with interface elements, and so on. While Android's official documentation requires apps to only use accessibility features to help people with disabilities interact with the app, accessibility is used by many apps to perform automated operations such as automatic check-in and giving "like". BIND_ACCESSIBILITY_SERVICE permissions are required to use accessibility. However, this permission is to ensure that only the system can bind to it. It is unlike the permissions declared in <uses-permission>. Therefore, the permission would not be notified to users that the application uses accessibility service when installing the application. To enable an accessibility feature, it typically requires the user to manually turn on the option in the system settings. Using a USB debug, the ADB command can also turn on the option without user awareness.

Third-Party Input Method: Android allows users to use input methods provided by a third-party application. However, a malicious third-party input method may record sensitive information entered by the user, such as private keys, mnemonics, and passwords. Vulnerabilities in third-party input methods can also widen the attack surface and threaten the confidentiality of user input.

Combine Attack Surface with Attack Vector

Based on our analysis, Table 1 summarizes the security consequences, that is, the violation of security properties, due to the wallet attack surface and the Android attack vectors, where **C** represents confidentiality, **I** for integrity, and **A** for availability, respectively.

Experiment

Based on the analyses in the previous section, this section presents attack experiments on two real-world cryptocurrency wallets, "Huobi Wallet" and "imToken." Although there are earlier works evaluating the security risks of mobile applications from the perspectives of the Android operating system and permission mechanism, they are different from our work on Android-based cryptocurrency wallets, which have different security goals. Hence, benchmarking with earlier works is not suitable.

Attack #1: Capture Sensitive Information from Screen Display

The main idea of this attack is to extract sensitive information displayed on the current screen by creating accessibility services. The detailed steps of the attack are as follows.

- Listen for screen changes: When the display content changes, accessibility events, TYPE_WINDOW_STATE_CHANGED or TYPE_WINDOW_CONTENT_CHANGED, are triggered. In the attack app, an accessibility service is built to monitor these two accessibility events from the wallet app by setting the accessibilityEventTypes field and packageNames field.
- Obtain displayed contents: Starting from the UI node that has display change events, we perform a depth-first traversal to get the text of all the contents on the changed interface.
- Sensitive information is identified from the obtained texts in accordance with the predefined rules (the rules shown in Table 2).

To show the experimental result, the obtained mnemonic and private key are displayed through a floating window on top of the application. If the displayed character is 40-bit hexadecimal or contains 12 words, it represents a private key or complete mnemonics. Thus, the attack is successful. As shown in Fig. 3, for both imToken and the Huobi Global Wallet, our attack can access the sensitive information using accessibility service.

No.	Message Type	Coin Type	Acquire Rule
1	Hexadecimal private key	BTC, ETH	Using regular expression "[0-9a-f]{64}"to match
2	WIF private key	BTC	Using regular expression "[5KL][1-9A-HJ-NP-Za-km-z]{50,51}" to match
3	Mnemonic	BTC, ETH	If a certain number of consecutive words come from the bip39 mnemonic lexicon, they are considered possible mnemonic
4	Keystore	ETH	Search key word ""crypto":{," acquire outer JSON

TABLE 2. Rules for sensitive information acquisition.

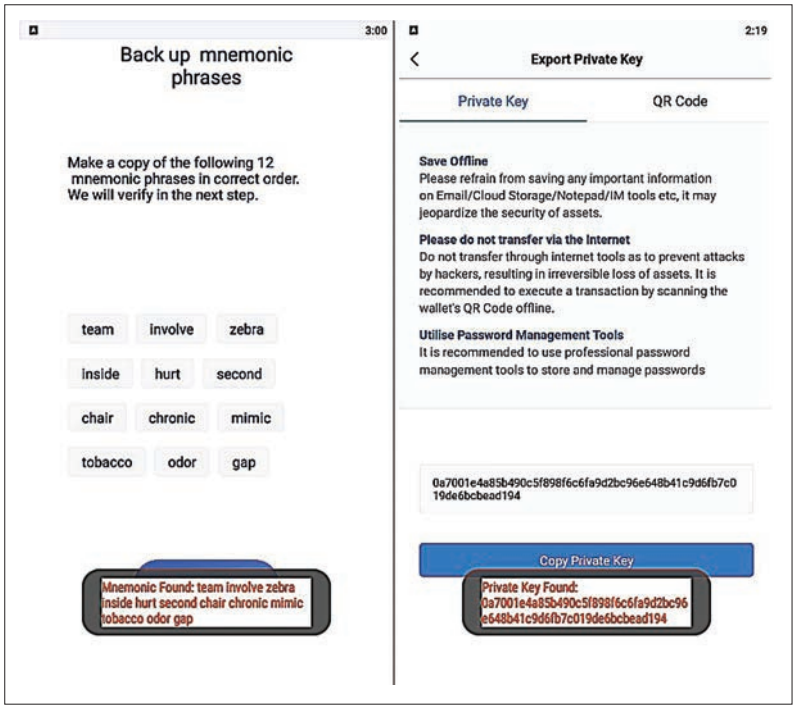

FIGURE 3. Result of Attack #1 Experiment.

Attack #2: Capture User Input via USB Debugging

The /dev/input/ folder contains various user input events for the Linux operating system, including input generated by touch screen, solid buttons, sensors, and so on. A typical Android app does not have permission to read the contents in the folder. However, the shell user performing USB debugging has such permission. Reading input files via USB debugging allows one to get keystroke inputs from a physical keyboard.

However, today's Android devices rarely have physical keyboards. For a virtual keyboard, which is more common nowadays, the attack app needs to determine user input by obtaining touchscreen events. For example, the /dev/input/event3 file contains various touchscreen events, each of which contains the corresponding touch coordinates on the screen. Combined with the layout of the virtual keyboard, the attack app can determine which key the user is pressing on the soft keyboard. If the attack app only targets for the sensitive input information, it needs to determine whether the user is currently conducting sensitive input, for example, entering the password via the soft keyboard. Also, the attack app needs to know the type of soft keyboard used. For the virtual keyboard of the common input method, the keyboard layout and the key positions are relatively fixed. It is easy to convert the coordinates into the corresponding keys. Some apps provide more secure input methods whose key positions are randomly changed every time users input. In such cases, screenshots, video recordings, or the accessibility features mentioned above can provide useful information about the timing and the keyboard layout.

The Huobi Wallet is such a wallet with its own random keyboard layouts. This secure keyboard is used when users need to enter a password in Huobi Wallet, which effectively prevents the third-party input method from accessing the password. The layout of the number keyboard is randomized. Huobi Wallet requires that the password should contain capital letters, lowercase letters, and numbers. The unknown layout of the number keyboard poses a great challenge to the attacker even if he knows every touch coordinate.

Huobi wallet's security keyboard does not trigger TYPE_WINDOW_STATE_CHANGED events when there is a click on the keyboard. However, by configuring the IncludeflagNotImportantViews in accessflagibility, changes in keyboard layout will trigger TYPE_WINDOW_STATE_CHANGED events. Therefore, by listening to this event, the attacker app can get the current keyboard layout, including the coordinate area for each key. At this point, if the attacker app can obtain the user's screen click positions through many ways, for example, USB debugging and suspension window, the exact input can be calculated by combining the keyboard layout and click positions.

Our attack app successfully obtained the layout information of the security keyboard of the Huobi Global Wallet via accessibility service. Combined with the background processes generated by USB debugging, the attack app successfully calculated the user's input password. The experiment results are shown in Fig. 4, which shows the input value and the value of the keyboard layout and means that the success of the attack.

In the first experiment, we used accessibility service to dynamically detect changes in the display content of the device and obtain UI node content to steal user privacy information. In our daily life, users may use a game plug-in in order to play phone games better, or use some apps to grab tickets. In these cases, accessibility service is needed. Of course, these apps cannot be obtained from official channels, and thus can be exploited by adversaries. In the second experiment, we successfully read the user input from a device linked with a computer, with USB debugging enabled. Often, these two scenarios can be seen in real life, which alerted us to study the current status of cryptocurrency wallet security.

Conclusion and Outlook

With the growing enthusiasm for cryptocurrencies based on blockchain technology, the security of crytpocurrency wallets is becoming more and more important. We have analyzed the risk of Android wallets from the perspective of wallet attack surface and Android features. From the experimental results of two popular cryptocurrency wallets, it shows that although cryptocurrency wallets provide different levels of security, there are still vulnerabilities that can be exploited by attackers.

In order to guarantee the security of cryptocurrency wallets on Android, it is important to reinforce the wallet with the system's new security features, prevent application supply chain contamination, and prevent repackaging attacks. In addition to Android cryptocurrency wallets, the security concerns of iOS cryptocurrency wallets, PC wallets, and hardware wallets are also non-neglectable, all requiring further research.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grants: U1936120, U1636216, 61762061, 61862042, 61762062); the National Key R&D Program of China (2017YFB0802805 and 2017YFB0801701); the Joint Fund of the Ministry of Education of China for Equipment Preresearch (No. 6141A020333); Jiangxi Key Laboratory of Smart City (No. 20192BCD40002); the Science and Technology Innovation Platform Project of Jiangxi Province (No. 20181BCD40005); the Fok Ying Tung Education Foundation of China (Grant 171058); and the Fundamental Research Funds for the Central Universities. Daojing He is the corresponding author of this article.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
[2] "Historical Snapshot — 10 Oct. 2019 CoinMarketCap," Dec 2019, https://coinmarketcap.com/ historical/ 20191011.
[3] M. Eminagaoglu *et al.*, "A Two-Factor Authentication System with Qr Codes for Web and Mobile Applications," *Proc. IEEE 5th Int'l. Conf. Emerging Security Technologies*, Alcala de Henares, Spain, 10–12 Sept. 2014, pp. 105–12.
[4] A. Kech, "Blockchain and Crypto: Will Security Issues Finally Be Dealt with in 2020?" https://cointelegraph.com/news/blockchain-and-crypto-willsecurity- issues-finally-be-dealt-with-in-2020.
[5] J. Cant, "Gatehub Crypto Wallet Data Breach Compromises Passwords of 1.4M Users," https://cointelegraph.com/news/gatehub-crypto-walletdata- breach-compromises-passwords-of-14m-users, 2019.
[6] T. Bui *et al.*, "Pitfalls of Open Architecture: How Friends Can Exploit Your Cryptocurrency Wallet," *Proc. 12th European Workshop on Systems Security*, Dresden, Germany, Mar. 25–28, 2019.
[7] E. Androulaki *et al.*, "Evaluating User Privacy in Bitcoin" *Proc. Int'l. Conf. Financial Cryptography and Data Security*, Okinawa, Japan, April 1–5, 2013, pp. 34–51.
[8] B. P. Sarma *et al.*, "Android Permissions: A Perspective Combining Risks and Benefits," *Proc. 17th ACM Symposium Access Control Models and Technologies*, Newark, New Jersey, USA, June 20–22, 2012, pp. 13–22.
[9] Y. Fratantonio *et al.*, "Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop," *Proc. IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 22-26 May 2017, pp. 1041–57.
[10] C. Ren *et al.*, "Towards Discovering and Understanding Task Hijacking in Android," *Proc. 24th USENIX Security Symposium*, Washington, D.C. USA, Aug. 2015, pp. 945–59.
[11] M. Xu, W. Sun, and M. Alam, "Security Enhancement of Secure USB Debugging in Android System," *Proc. 12th Annual IEEE Consumer Commun. Networking Conf.*, Las Vegas, NV, USA, 9–12 Jan. 2015, pp. 134–39.
[12] R. Mayrhofer, "When Users Cannot Verify Digital Signatures: On the Difficulties of Securing Mobile Devices," *Proc. IEEE 10th Int'l. Conf. High Performance Computing and Commun.*, Zhangjiajie, China, 13–15 Nov. 2013, pp. 1579–84.
[13] Z. Xu, K. Bai, and S. Zhu. "TapLogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors," *Proc. ACM Conf. Wireless Network Security (WiSec)*, Tucson, AZ, USA May 2012, pp. 113–24.
[14] W. Diao *et al.*, "No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis," *Proc. IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 22–26 May 2016, pp. 414–32.
[15] F. Roesner *et al.*, "User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems," *Proc. IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 20–23 May 2012, pp. 224–38.

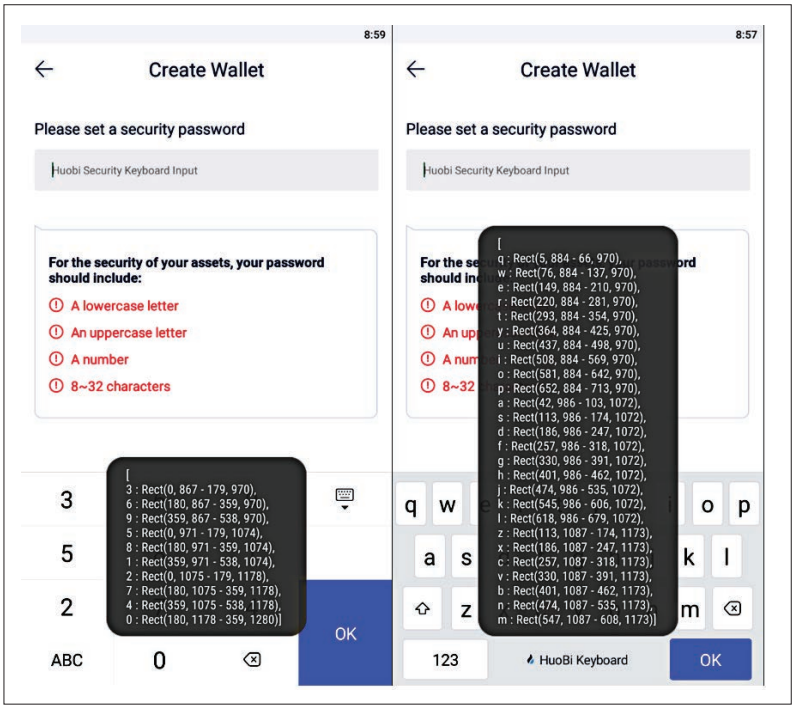

FIGURE 4. The result of attack #2 Experiment.

Biographies

Daojing He [S'07, M'13] received the B.Eng. (2007) and M. Eng. (2009) degrees from Harbin Institute of Technology (China) and the Ph.D. degree (2012) from Zhejiang University (China), all in computer science. He is currently a professor in the School of Software Engineering, East China Normal University, P.R. China. His research interests include network and systems security. He is on the editorial board of several international journals such as *IEEE Communications Mag*.

Shihao Li received the M.Eng. degree from the School of Software Engineering, East China Normal University, P.R. China.

Cong Li is currently a master student in the School of Software Engineering, East China Normal University, P.R. China.

Sencun Zhu is an associate professor in the Department of Computer Science and Engineering at Penn State University (PSU). He received the B.S. degree from Tsinghua University, the M.S. degree from the University of Science and Technology of China, and the Ph.D. degree from George Mason University in 1996, 1999, and 2004, respectively. His research interests include wireless and mobile security, software and network security, and fraud detection. He is the editor-in-chief of *EAI Transactions on Security and Safety*, and an associate editor of *IEEE Transaction on Mobile Computing*.

Sammy Chan [S'87, M'89] received his B.E. and M.Eng. Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and a Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. Since December 1994 he has been with the Department of Electrical Engineering, City University of Hong Kong, where he is currently an associate professor.

Weidong Min [M'12] received the B.E., M.E. and Ph.D. degrees in computer applications from Tsinghua University, China in 1989, 1991 and 1995, respectively. He is currently a professor and the Dean, School of Software, Nanchang University, China. He is an Executive Director of the China Society of Image and Graphics. His current research interests include image and video processing, artificial intelligence, big data, distributed systems and smart city information technology.

Nadra Guizani is a clinical assistant professor and a Ph.D. candidate at the School of Electrical and Computer Engineering at Purdue University. Her research interests include machine learning, mobile networking, large data analysis, and prediction techniques. She is an active member of both the Women in Engineering program and the Computing Research Association.