An Attack to One-Tap Authentication Services in Cellular Networks

Zhiwei Cui, Baojiang Cui, Junsong Fu, and Bharat K. Bhargava, *Life Fellow, IEEE*

***Abstract*— The One-Tap Authentication (OTAuth) based on the cellular network is a password-less login service provided by Mobile Network Operator (MNO) through the unique communication gateway access technique. The service allows app users to quickly sign up or log in with their mobile phone numbers without entering a password. Due to its convenience, OTAuth has been widely used by various apps. However, some studies have elaborated that OTAuth services are of great drawbacks from the perspective of mobile security and identified several flawed designs, which make the MNO cannot distinguish malicious apps from normal ones and cause impersonation attacks. In this paper, we further analyze OTAuth services from the perspective of 4G and 5G cellular networks and focus on two important procedures in which the cellular network plays an important role in OTAuth services. Not surprisingly, we discover a new fundamental design flaw in determining whether the runtime environment supports OTAuth services. Moreover, we propose a mature attack paradigm by exploiting this flaw, which allows an attacker to login or register one app as a victim. To evaluate the impact of the attack, we have examined 100/90/100 Android/iOS/HarmonyOS apps for OTAuth services of 3 mainstream MNOs in China. The experimental results show that our proposed attack is applicable to almost all the apps that support OTAuth services, and affects more apps than the attacks that have been reported before. Finally, we propose several countermeasures to defend against the attack. Note that, for security's sake, we have already reported our findings to authorized parties and received their confirmations.**

***Index Terms*— One-Tap authentication, network security, cellular network.**

I. Introduction

With the proliferation of various mobile network services, it is imperative to provide a convenient and secure authentication process. The traditional password-based authentication scheme requires app users to enter complex passwords correctly. Each user needs to remember different passwords for various apps, which would disturb users and degrade the user experience. Apparently, the complicated registration process greatly affects the user experience and it hinders the increase of the number of app users. To get rid of this dilemma, app developers and the Mobile Network Operator (MNO) have proposed the mainstream Short Message Service (SMS) based authentication in the form of business cooperation. App developers pay a certain fee to MNOs for this authentication service to enable app users to register or log in to the app account with their mobile phone numbers, thus eliminating the need for app users to remember multiple passwords. The app users need to click on the specified location on the phone screen to request a verification code, and then enter the code received in a few seconds to log in to their accounts. Statistically, it takes about 16s to log in to the app using the SMS-based authentication scheme which is not convenient enough. Meanwhile, numerous studies have shown that the verification codes are easily stolen by attackers to bypass authentication [1], [2], [3]. To address the shortcomings of the above two authentication schemes at the same time, the cellular network based One-Tap Authentication (OTAuth) as a new authentication scheme has rapidly emerged.

OTAuth is a successful application of Mobile Connect, which is a secure universal mobile device based digital identity service proposed by the Global System for Mobile Communications Association (GSMA) [4]. OTAuth is a password-less login service provided by MNO and has been adopted by more than 70 MNOs around the world [4]. During the process of using OTAuth services, the mobile phone must use cellular network (4G/5G), not a Wi-Fi network. Then the MNO uses its unique communication gateway access technique to identify the user's mobile phone number accurately. Specifically, the traffic sent by an app client to request OTAuth services will flow through the gateway of the cellular network managed by the MNO. The MNO can access the gateway to accurately determine the phone number of the mobile user to which the cellular network traffic belongs. By matching the users to their mobile phone numbers, the MNO can provide secure authentication services for apps. With OTAuth services, app users can sign up or log in to the app using their mobile phone numbers as accounts without entering a password.

Compared with traditional schemes, OTAuth services have the advantages of simple operation, short waiting time, and high security. The app user only needs to tap once on the screen (the button marked by the red box in Fig. 1) to login into one app. The process takes only about 1.5s. And the communication gateway access technique effectively avoids the possibility of mobile phone numbers being intercepted at the application layer. In addition, the OTAuth service fee that app developers need to pay is also much cheaper than the

Manuscript received 6 January 2023; revised 1 August 2023; accepted 2 August 2023. Date of publication 14 August 2023; date of current version 21 August 2023. This work was supported by the National Natural Science Foundation of China under Grant 62001055. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Valeria Loscri. *(Corresponding author: Baojiang Cui.)*

Zhiwei Cui, Baojiang Cui, and Junsong Fu are with the School of Cyberspace Security and National Engineering Laboratory for Mobile Network Technologies, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zwcui@bupt.edu.cn; cuibj@bupt.edu.cn; fujs@bupt.edu.cn).

Bharat K. Bhargava is with the Department of Computer Science, Purdue University, West Lafayette, IN 47906 USA (e-mail: bbshail@purdue.edu).

Digital Object Identifier 10.1109/TIFS.2023.3304840

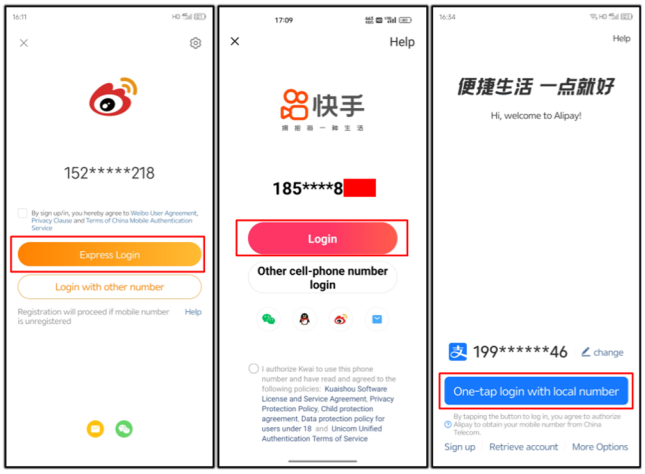

Fig. 1. Examples of OTAuth interfaces in apps supported by three MNOs in China.

SMS-based authentication. Given that the OTAuth service has so many advantages, it is widely used by various apps, such as Sina Weibo, Alipay, TikTok, etc.

However, while OTAuth provides app users with convenient login services, it also exposes them to a wide range of security risks. For example, OTAuth SDKs provided by MNOs to apps may have insecure implementation. Due to the large number of apps using the service, any security issues will involve hundreds of millions of users and cause a huge negative impact. It is hence important to investigate the design and deployments of cellular network based OTAuth services for identifying potential vulnerabilities.

In recent years, several researchers have analyzed the security of OTAuth services in detail and also identified a set of flaws. The attack against OTAuth services was first publicly demonstrated at GeekPwn2019 [5]. Researchers launched an app authentication bypass attack by connecting the victim's hotspot. Zhou et al. [6] have conducted the first comprehensive investigation against OTAuth services and proposed two attack scenarios. Both of the above studies perform security analysis and attacks from the perspective of mobile security, such as hooking specific methods [7] and installing malicious apps to obtain tokens. These attack methods have the following limitations: (1) The attacker's phone needs to be running in a rooted environment, which is difficult or impossible for most phones, especially iPhones running iOS; (2) Some apps have safeguards to check whether functions are hooked, which makes these attacks no longer applicable; (3) It is very difficult to trick victims into installing malware on their phones. In addition, the protection measures proposed to enhance the security of OTAuth services focus on mobile security. However, the cellular network that empowers OTAuth services has not received much attention from researchers.

In this paper, we further investigate the security of OTAuth services from the 4G and 5G cellular networks perspective. We have analyzed OTAuth services provided by three MNOs in China, namely China Mobile, China Unicom, and China Telecom. Specifically, we focus on two key procedures involving the cellular network in OTAuth services: (1) MNOs recognize the phone number through the communication gateway access technique. (2) OTAuth SDK embedded in apps detects whether the network status supports OTAuth. We have proposed an attack exploiting the new fundamental design flaw of OTAuth we found: ***During the process of detecting the network status, OTAuth SDK only checks the parameters of the connected MNO, and cannot judge the authenticity of the currently connected MNO***. Our attack does not have the limitations of already disclosed attacks. In our attack, an attacker only needs to connect to the Wi-Fi hotspot established through the cellular network of the victim's phone. Then, the attacker can bypass the OTAuth scheme and use the victim's phone number to register or login into various app accounts, which will lead to serious privacy leakage.

To ensure that the proposed attack is indeed realizable in practice, we have conducted a measurement against several popular apps in the mobile app market, including 100 Android apps, 90 iOS apps, and 100 HarmonyOS apps. In addition, we have tested OTAuth services provided by 3 MNOs for each app. Among these apps, 73 Android apps, 60 iOS apps, and 72 HarmonyOS apps support the OTAuth service. Although OTAuth services of the 3 MNOs are somewhat different, our attack can be successfully carried out and affect more than 1.6 billion subscribers [8]. The results show that our proposed attack affects almost all apps that support OTAuth services. For a few apps with additional verification for new devices, attackers can only impersonate the victim to register an account in advance. Our attack affects a larger number of apps than the previously reported attacks [5], [6]. Unlike them, in our proposed attack, the attacker's phone does not need to be rooted, which will bypass the app's security checks on the running environment. Moreover, previous attack approaches require extracting some app information from each target app. However, our attack does not require additional work for different apps.

We have conducted an informal survey to assess the feasibility of the precondition for our attack (i.e., sharing the victim's cellular network). According to the statistical results, we analyzed some scenarios where the attack is most likely to occur. In addition, we have identified some other insecure implementations, such as overly simple parameter checking, too few digits of masked mobile phone number, and remote login without alerting users. Finally, we propose several potential countermeasures for these problems to enhance the security of OTAuth services.

To summarize, we make the four following contributions:

- We found a new design flaw and several implementation weaknesses in OTAuth, which is a new authentication scheme provided by MNOs and has been adopted by various apps.
- We designed an attack exploiting the fundamental design flaw of OTAuth services. In our attack, an attacker can bypass the authentication and impersonate the victim to perform actions to apps.
- We performed a measurement to assess the feasibility and impact of our proposed attack. The results showed that almost all apps supporting OTAuth services are vulnerable to our attack. Additionally, our attack is more effective than the previously reported attacks.

- We proposed recommended solutions to eliminate security issues in OTAuth services.

Ethical Considerations: We conducted all the experiments in a responsible manner. First, only the authors' phone numbers and app accounts were involved in the experiment. It hence did not affect other users. Second, the radio transceiver and all tested phones are put into a radio-isolated shield box to prevent the exposure of signals. We have reported our findings to the China National Vulnerability Database (CNVD) to inform these three affected MNOs and obtained three high severity vulnerability numbers (CNVD-2022-47255, CNVD-2022-47256, and CNVD-2022-47257), which are scored 8.3 out of 10 in CVSS 2.0.

The rest of this paper is organized as follows. We summarize the related work in Section II. The OTAuth services and the threat model are presented in Section III. We then thoroughly analyze the security of OTAuth services in Section IV. Based on the drawbacks of OTAuth services, a mature attack paradigm is provided in Section V. We then evaluate the performance of our scheme in Section VI. A survey is provided in Section VII to illustrate the seriousness of our attack to OTAuth services. Moreover, some other weaknesses are also discussed in Section VIII and a set of countermeasures are provided in Section IX. At last, we conclude this paper in Section X.

II. Related Work

In this section, we introduce the security research related to two services for mobile authentication provided by MNOs (i.e., SMS-based authentication and OTAuth) and bypassing authentication on apps.

A. SMS-Based Authentication

The security issues of SMS-based authentication service are hot research topics in recent years. The authentication service uses the One-Time verification code contained in the SMS to verify the identity. The security assumption of this service is that only the user with the target phone number can receive this SMS. However, once the SMS is leaked, the attacker may log into the victim's account. Attackers can install malware on victims' phones to steal the SMS without their awareness [9], [10]. Enck et al. [11] conducted research on DoS attacks by abusing SMS. Liu et al. [12] showed that an attacker can use the side channel method to replicate a USIM card to receive the SMS as a victim. Attackers can also use the USIM swap attack [13] to deceive the MNO to transfer the victim's phone number to a new USIM card. It is well known that GSM (2G) is insecure as attackers can obtain the SMS through the MitM attack or passive sniffing [2]. Zhang et al. [14] present the first large-scale characterization of fake base station (FBS) spam ecosystem and point out that the abused attack has caused huge economic losses and privacy leakage to users. While a user can choose a safer 3G/4G/5G environment, by exploiting the vulnerabilities in redirection procedure, an attacker can force a target user's device to fall back to the vulnerable GSM network [15]. To solve this, Android 12 allows users to disable the GSM network connections to prevent the FBS attacks [16]. However, the implementation vulnerabilities of the IP Multimedia Subsystem (IMS) in 4G networks can also be exploited to devise the SMS spoofing attack [17]. Cui et al. [18] analyzed voice services in 5G networks and designed the SMS spoofing and interception attack, which can be exploited to log into the victim's app account.

B. OTAuth and Its Vulnerabilities

Since SMS-based authentication faces various security threats, MNOs introduce OTAuth to provide safer and more convenient authentication services. However, some recent studies have shown that OTAuth still has security risks. OTAuth services can be applied to a website or app authentication. When OTAuth is used for website authentication, it can be abused to collect phone numbers of users who have browsed specific web pages, thereby enabling targeted advertising [19]. This poses a great threat to user privacy.

When providing authentication for apps, the researchers successfully implemented the app authentication bypass attack by exploiting the vulnerability of OTAuth services at GeekPwn2019 [5]. Zhou et al. [6] conducted the first systematic security analysis of OTAuth services and proposed two attack methods. In addition, they carried out a large-scale measurement study on a set of popular apps to better understand how apps are affected. They confirmed that a large portion of apps support OTAuth services and these apps could be affected by their attacks. They randomly selected some ones from these apps for attack verification. Correspondingly, they proposed countermeasures to protect the token by using the operating system (OS). However, the above security solutions are all from the perspective of mobile security, and cannot effectively protect the attack proposed from the perspective of cellular networks in this paper.

C. Bypassing Authentication on Apps

Recently, the security analysis in authentication services on apps has gained a lot of traction. The attack effects of these researches are similar to ours, that is, to bypass the authentication on apps to perform malicious operations on these apps as the victim. The difference from our works is that these researches focus on authentication services provided by apps, rather than MNOs. Wang et al. [20], [21] conducted a security analysis on a large number of apps using the OAuth-based authentication scheme and found various vulnerabilities. Park and Yi [22] demonstrated that an attacker can bypass the authentication mechanism of a mobile messenger app to not only view past or real time conversations but to also illegally use of pay items. Song et al. [23] developed VPDroid, a transparent Android OS-level virtualization platform, to test app's auto-login function and successfully implemented the clone attack. In addition, researchers discovered that Android password manager apps can be bypassed without rooting the mobile phone to get the premium version of an Android app for free [24]. Bianchi et al. [25] focused on the authentication schemes based on the device-public information, which consists of properties and data that an app can obtain from a device, and could easily hijack the victim's account,

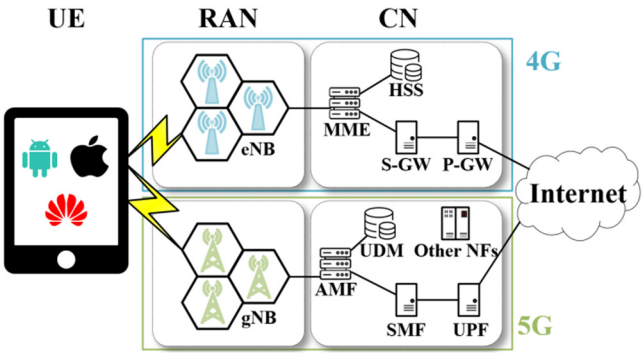

Fig. 2. The architectures of 4G and 5G networks.

steal private information, and send or receive messages on behalf of the victim. Rupprecht et al. [26] proposed a novel DNS spoofing attack exploiting the basic protocol flaw of Long Term Evolution (LTE), which can be used to steal the username and the password of an app account. Wang and Wang [27] systematically explored the failure of security proofs of multi-factor authentication schemes. To enhance the security of authentication in services, some new protocols are designed, such as an asymmetric protocol against pre-computation attacks [28] and a secure user authentication scheme with lightweight computing for cloud-assisted Internet of Things (IoT) [29].

III. OTAuth Service and Threat Model in Cellular Networks

In this section, we first present the structure of cellular networks and then discuss the flow of OTAuth services in detail. Finally, we define the threat model. These contents are the foundation of analyzing the security of OTAuth services and designing our attack paradigm.

A. 4G and 5G Network Architectures

As shown in Fig. 2, the network architectures of 4G and 5G, which are mainstream cellular networks, contain three main components: the User Equipment (UE), the Radio Access Network (RAN), and the Core Network (CN). We briefly introduce the functionalities of these components.

1) UE: The UE is a device providing data transmission for users through the cellular network, such as the mobile phone and the IoT device. Each UE has a unique International Mobile Equipment Identity (IMEI). The UE holds a Universal Subscriber Identity Module (USIM) card with a unique Integrate Circuit Card Identity (ICCID). The USIM stores the International Mobile Subscriber Identity (IMSI) uniquely identifying each user and other information, such as the authentication key. In our research, we focus on mobile phones with different operating systems that can run apps supporting OTAuth services.

2) RAN: The RAN is a radio access network composed of base stations (i.e., Evolved NodeB (eNB) in 4G and next Generation NodeB (gNB) in 5G). The geographic area is divided into hexagonal cells, each of which is served by at least one base station. Therefore, the 4G and 5G networks are called cellular networks. The RAN acts as a bridge for the communication between the UE and the CN. A UE always chooses the base station with the strongest signal. The base station has two important parameters related to this paper: the Public Land Mobile Network (PLMN) and the frequency of the downlink wireless channel (i.e., E-UTRA Absolute Radio Frequency Channel Number for downlink (DL_EARFCN) in 4G and NR Absolute Radio Frequency Channel Number for downlink (DL_NR_ARFCN) in 5G). The PLMN consists of the Mobile Country Code (MCC) and the Mobile Network Code (MNC). For example, the PLMN of China Mobile is 46000 (MCC: 460, MNC: 00). The eNB of the malicious 4G network to be built in our research needs to keep the PLMN and the DL_EARFCN consistent with the legal eNB of the target MNO.

3) CN: The CN is responsible for providing mobility and session management to subscribers. The CN contains various network functions (NFs) and communicates with the Internet. The Home Subscriber Server (HSS) in 4G (similar to the Unified Data Management (UDM) in 5G) stores the same parameters as those in the USIM card, such as the IMSI and the authentication key. The Mobility Management Entity (MME) in 4G (similar to the Access and Mobility Management Function (AMF) in 5G) uses these parameters to complete the mutual authentication with the UE through the Authentication and Key Agreement (AKA) procedure. And the Packet Data Network GateWay (P-GW) in 4G (similar to the User Plane Function (UPF) in 5G) assigns IP addresses to UEs and provides data services. Therefore, the MNO can judge which user the cellular data belongs to through these two NFs that are the gateways of the 4G and 5G core networks respectively. The UE can enjoy data services after setting the data traffic Access Point Name (APN). For example, the APN of China Mobile is cmnet. In addition, the CN also has the important parameter PLMN. In our research, these two parameters of the malicious CN need to be consistent with the target MNO.

B. An Introduction of OTAuth Services

OTAuth is a third-party based authentication service provided by MNOs. For an app that supports OTAuth services, a user can log in to an app account with the user's phone number with just one tap. If the phone number has not been registered for an app account, most apps will automatically create a new account associated with the phone number. As shown in Fig. 1, the app displays the masked phone number to request authorization from the user. It is worth noting that the number of digits the phone number is masked with is different, which will lead to some security issues (see Section VIII). The user can choose to log in this way without entering the username and the password, or opt to log in with another app account (similar to performing Single-Sign-On [30] with Google) [6].

The phone number that uniquely identifies each user can be used as an app account. For privacy protection, the app cannot directly obtain the phone number by requiring the system permissions, such as READ_PHONE_NUMBERS. Based on the MNO's communication gateway access technique, the OTAuth service provides another method for apps to obtain

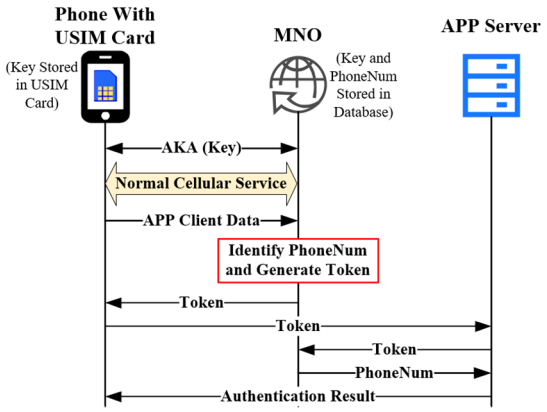

Fig. 3. The model of the OTAuth service.

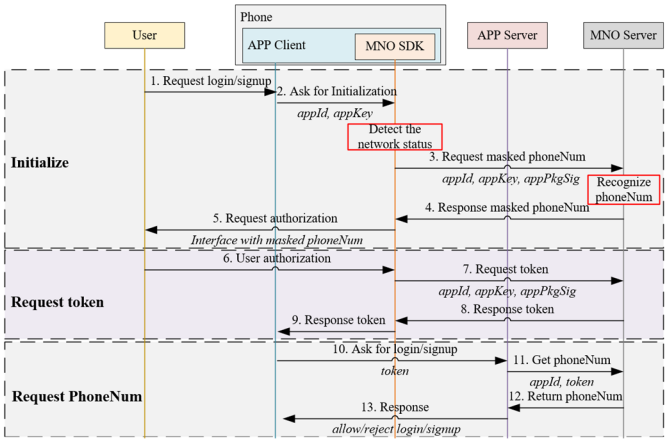

Fig. 4. The protocol flow of OTAuth [6].

the phone number that is using the cellular network. In the authentication process, the MNO acts as a third party that is trusted by both the cellular network user and the app server.

Fig. 3 shows the basic model of the OTAuth service. Before using the OTAuth service, the user's mobile phone needs to be connected to the MNO's cellular network. To do this, the mobile phone equipped with a valid USIM card assigned to the user by the MNO needs to pass the secure AKA authentication before registering to the MNO's network [31], [32], [33], [34]. After the AKA phase, a secure connection is established between the user's mobile phone and the MNO. In this way, the user's mobile phone can safely consume cellular services and the traffic sent by the mobile phone will flow through the gateway managed by the MNO. In addition, since the MNO will assign a local area network (LAN) IP to the mobile phone, the MNO can access the gateway to determine which user the traffic data belongs to according to the IP, and then obtain the user's phone number. Since the authentication key stored in the USIM card is not accessible, the attacker cannot spoof the victim's access to the network.

Specific to OTAuth services, the app client sends the specified data to the MNO's server through the cellular network. The MNO's server identifies the phone number to which the data traffic belongs. It generates a token associated with the phone number and sends the token to the app client. Then the app client sends the token to the app server. The app server forwards the token to the MNO's server to obtain the associated phone number. In this manner, the app server can determine whether to allow this login or sign-up request based on the phone number. From the above process, it can be seen that there are two requirements for using the OTAuth service: (1) the app should introduce the MNO's OTAuth service. (2) the mobile phone should be connected to the cellular network and enable mobile data.

Given the convenience and security of the OTAuth service, many popular apps have adopted this authentication service. Among the apps we have tested, more than 70 of them support OTAuth services. According to statistics, China Mobile's OTAuth service has been called more than 1.86 trillion times as of September 2022 [35]. The app introduces OTAuth services by integrating the OTAuth SDK. There are two different types of OTAuth SDKs, including the official SDK developed by MNO and the third-party SDK. All three MNOs in China (China Mobile, China Unicom, and China Telecom) provide the SDKs, each of which supports OTAuth services for the other two MNOs. For example, an app can use the SDK made by China Mobile to provide OTAuth services to subscribers of the two other MNOs. The third-party SDK (e.g., Jiguang [36], Shanyan [37]) integrates the functions of all MNOs' SDKs and provides more friendly APIs for app developers to integrate OTAuth services.

C. Threat Model

In order to assess the security of OTAuth services, a realistic threat model is necessary. Inspired by the existing work [27], [28], [29], the attacker's capabilities are as follows: (1) The attacker cannot obtain key information stored in the victim's USIM card (e.g., the key and the IMSI) to impersonate the victim to access the MNO. (2) The attacker can access the hotspot established by the victim over the cellular network. (3) The attacker cannot control OTAuth server provided by the MNO. (4) The attacker cannot install a malicious app on the victim's phone to steal sensitive data. (5) The attacker has a fully controlled mobile phone to analyze OTAuth services, for example, to capture the OTAuth service traffic. (6) The attacker has a certain basic knowledge of cellular networks and can build a cellular network for experimentation.

IV. Systematic Analysis of OTAuth Security

In this section, we first analyze the protocol of the OTAuth service in detail. Then, we present the existing security analysis and attack schemes of OTAuth. The shortcomings of existing work are also discussed. At last, we reanalyzed the OTAuth and find a new vulnerability of OTAuth. Based on the analysis in this section, an attack paradigm will be provided in the next section.

A. The Protocol Flow of OTAuth

As shown in Fig. 4, the protocol flow of the OTAuth service mainly involves five entities, namely the user, the app client, the MNO's SDK, the app server, and the MNO server. The user of the mobile network is a subscriber with a legitimate USIM card assigned by the MNO. The app client containing the OTAuth SDK made by the MNO can provide users with

the OTAuth service interaction interface. The MNO server can identify the user's mobile phone number based on the cellular traffic, and the app server determines whether to allow the user's registration or login request according to the mobile phone number obtained from the MNO server. To introduce OTAuth services into an app, the app developer first needs to request the MNO to assign *appId* and *appKey*, which can uniquely identify the app. In addition, the app developer should provide the MNO with the IP of the app server. Then the app developer embeds the MNO's OTAuth SDK into the app client and configures these parameters.

The protocol flow of OTAuth services can be mainly divided into three phases (i.e., the initialization phase, token request phase and phone number request phase) [6]. We discuss the above three phases respectively in the following.

1) Initialization Phase: In the initial phase, the app client requests the masked phone number from the MNO server to display it on the authorization interface. First, the user clicks the login (or sign-up) button to trigger OTAuth flow (step 1). After receiving the user's request, the app calls the interface provided by the MNO's SDK with *appId* and *appKey* as parameters to ask for initialization (step 2). Then the MNO's SDK detects whether the network status supports OTAuth. Specifically, the SDK determines whether the mobile phone has access to the MNO's cellular network and enables mobile data through specific methods (e.g., the functions of *android.net.ConnectivityManager.getActiveNetworkInfo* and *android.telephony.TelephonyManager.getSimOperator* provided by the system).

Note that, if the mobile phone has opened both Wi-Fi and mobile data at the same time, the SDK will automatically switch to mobile data to transfer the information related to OTAuth services. If the network is as expected, the SDK collects *appPkgSig* (the fingerprint of the signing certificate [38]) through the API *getPackageInfo*. It then sends some parameters (including *appId*, *appKey*, and *appPkgSig*) to the MNO server for the masked phone number (step 3). After confirming that these parameters are legitimate, the MNO server identifies the user and returns the user's masked phone number to the SDK (step 4). Finally, the SDK pulls up an interface as shown in Fig. 1 to request the user's authorization (step 5).

2) Token Request Phase: In this phase, the app client requests a token associated with the phone number, *appId*, and *appKey*. And in the next stage, the app server can obtain the user's phone number through the token.

In this process, the user first approves OTAuth services by clicking the button (marked by the red box in Fig. 1) on the authorization interface (step 6). Then the MNO's SDK requests the token through the interface provided by the MNO server (step 7). The parameters of this interface include *appId*, *appKey*, and *appPkgSig*. It is worth noting that this request is transmitted through the cellular network. After these parameters are verified, the server will generate a token associated with the user's phone number and return it to the app client (step 8 and step 9). Note that, users do not send their phone numbers to the MNO and this is reasonable considering that the MNO always knows the connected subscribers. This is the core smart trick of OTAuth.

3) Phone Number Request Phase: In the last phase, the app server requests the user's phone number from the MNO server and decides whether to allow the user's login or sign-up request through the phone number. In this process, the app client first sends the token obtained in the second phase to the app server to ask for login or sign-up (step 10). Note that, the app client is not forced to send this message over the cellular network and the information can be transmitted through the Wi-Fi. After receiving the authentication request from the app client, the app server sends the received token and *appId* to the MNO server to obtain the user's phone number (step 11). After receiving the request from the app server, the MNO server will check whether the IP of the app server is previously configured, and then judge whether the token and *appId* are consistent with each other. If these statements are true, the MNO server will return the phone number to the app server (step 12). Lastly, the app server determines whether to approve the user's login or sign-up request based on mobile phone number (step 13).

B. Previous Security Analysis of OTAuth Services

The security of OTAuth services deployed in China has been discussed in previous works [6]. The researchers identified a fundamental design flaw in the OTAuth service: The MNO server is inability to effectively verify whether the authentication request is made by a legitimate app or a malicious app. The researchers pointed out that the root cause of the flaw is the app's inability to securely use the mobile identity. Note that the design of OTAuth services ignores the involvement of the operating system. In this case, the MNO cannot distinguish different apps on the same mobile phone. Moreover, since the key information (i.e., *appId*, *appKey*, and *appPkgSig*) of a legitimate app is stored in plain-text in a phone, it is easy for an attacker to obtain these parameters and then build a malicious app.

Based on the analysis results, the researchers proposed two attack scenarios to obtain an effective token (identified as $token_V$) associated with the victim's phone number, which can be used to bypass authentication. Then the attacker performs the normal OTAuth flow of the target app on the attacker's own phone. After this, the MNO will send a token (identified as $token_A$) associated with the attacker's phone number to the app client in the attacker's phone. Since the attacker has full control of the attacker's phone, the attacker is able to prevent the app client from sending the $token_A$ to the app server. The attacker then replaces the $token_A$ with the previously obtained $token_V$ to bypass the authentication of the app server. Since the valid $token_V$ is associated with the victim's phone number, the app server can request the phone number from the MNO's server through the $token_V$. In this way, the app server will mistakenly believe that the attacker is a legitimate user and authorize the attacker's login or registration request.

In the first attack scenario, the attacker installs a malicious app on the victim's phone. More specifically, the attacker first builds the malicious app, which hard-codes the *appId*, *appKey*, and *appPkgSig* of the target app. Then, once the malicious app

successfully runs on the victim's phone, it sends some data related to the target app through the victim's cellular network and obtains a valid token. In the second attack scenario, the attacker's phone needs to connect to the mobile hotspot of the victim's phone. The attacker also needs to create an identical malicious app that is installed on the attacker's own phone, not the victim's phone. The malicious app sends data through the victim's hotspot to request a valid token. In this attack scenario, the OTAuth SDK checks the network status through specific methods and forces the data to be sent over the cellular network, which contradicts requesting authentication through the victim's hotspot. To solve this problem, researchers use the hook technique to overload these methods to return the correct condition each time the check is performed.

C. Shortcomings of Previous Security Analysis

First, the above security analysis is conducted from the mobile security perspective, and the countermeasures proposed by the researchers are also security enhancements to the app. However, the researchers overlooked the impact of cellular networks on the security of OTAuth services.

Second, under the two attack scenarios, an attacker has a chance to obtain a legitimate token, and then they need to hook and replace the token sent by the legitimate app on the attacker's phone. It can be observed that the attack chains of the above two attacks are too long and as a result, the attack effect decreases. The above two attack approaches have at least one of the following limitations: (1) The attacker needs to extract some information about each target application to build a malicious app. And extra work may be required for some apps that introduce app hardening techniques (e.g., the code obfuscation and the packing). (2) In the first attack, the core challenge is that the attacker needs to install the malicious app on the victim's mobile phone. To our knowledge, this is of great difficulty in real life especially for the new smartphones with more secure OS systems (e.g., the victim's phone is an iPhone). On iPhone, all apps should be obtained from the app store and protected by the sandbox [39]. The user can access these apps on their iPhones without undue fear of malicious apps. (3) The attacker needs a rooted mobile phone, which can be fully controlled. On the rooted mobile phone, the hook technique is exploited to perform key malicious actions (e.g., replace tokens, tamper with the OTAuth SDK, and bypass the check on cellular network status) necessary for attacks. However, some apps will prohibit OTAuth services if they detect that the mobile phone is in the rooted runtime environment.

D. Our Security Analysis of OTAuth Services

Although some MNOs in other countries also have OTAuth services [4], we are unable to obtain USIM cards of these MNOs to conduct the testing for OTAuth services due to geographical restrictions. Therefore, we mainly focus on the OTAuth services provided by the three MNOs in China in this paper. However, we hope that our research can inspire researchers in other countries to conduct related studies.

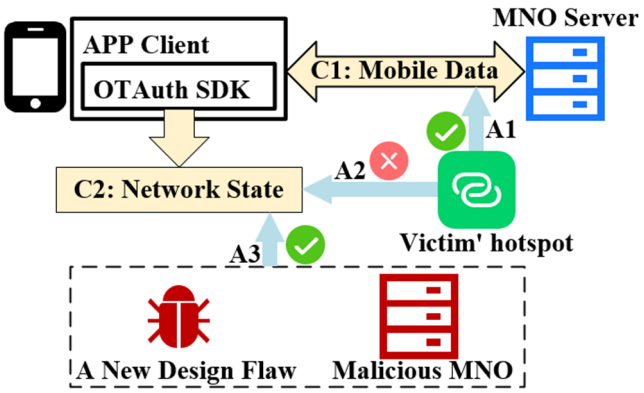

Fig. 5. Security analysis for the two key procedures involving the cellular network in OTAuth services.

Based on the previous work in [6] and the observations that the work is insufficient, we further analyze the security of OTAuth services by focusing on the cellular network which plays an important role in OTAuth services and provide a more mature attack paradigm introduced in the next section. Fig. 5 shows our analysis results. Specifically, we analyze the two key procedures involving the cellular network in OTAuth services: (1) MNOs identify the phone number through the mobile data (identified as C1 in Fig. 5); (2) OTAuth SDK detects whether the network state supports OTAuth services (identified as C2 in Fig. 5). If these procedures are flawed, an attacker could impersonate a victim to log into apps with OTAuth services. Unfortunately, our results show that these two procedures indeed have vulnerabilities. Our analysis can be divided into three aspects: (1) A1: Bypass C1 with the victim's hotspot. (2) A2: Contradiction between C2 and A1. (3) A3: Resolve the contradiction and bypass C2 based on a new design flaw and the help of a malicious MNO. We elaborate on the above three aspects respectively in the following.

A1. The MNO server obtains the mobile phone number through the unique gateway access technique. If the attacker wants to trick the MNO server to consume OTAuth services as the victim, it requires the OTAuth SDK in the attacker's phone to communicate with the MNO server via the victim's cellular network channel. This can be achieved by sharing the victim's hotspot established through the cellular network of the victim's phone. Through analysis, we found that this attack is of great difficulty to be prevented because of the inherent property of the hotspot service provided by almost all the phones. As a consequence, this is a wise manner to build the attack chain based on the victim's hotspot.

A2. After connecting directly to the victim's hotspot, the network type of the attacker's mobile phone is Wi-Fi. However, in the initial stage of OTAuth services, the OTAuth SDK will check the network status (i.e., network type and MNO's parameters) of the mobile phone and require the phone to access the cellular network. To resolve this contradiction, it is necessary to deceive the SDK that the network status is expected. Since the SDK obtains the network status through the OS interface, previous research works [5], [6] hook these interfaces and modify the network status on the phone to spoof the SDK. The precondition of this method is that the

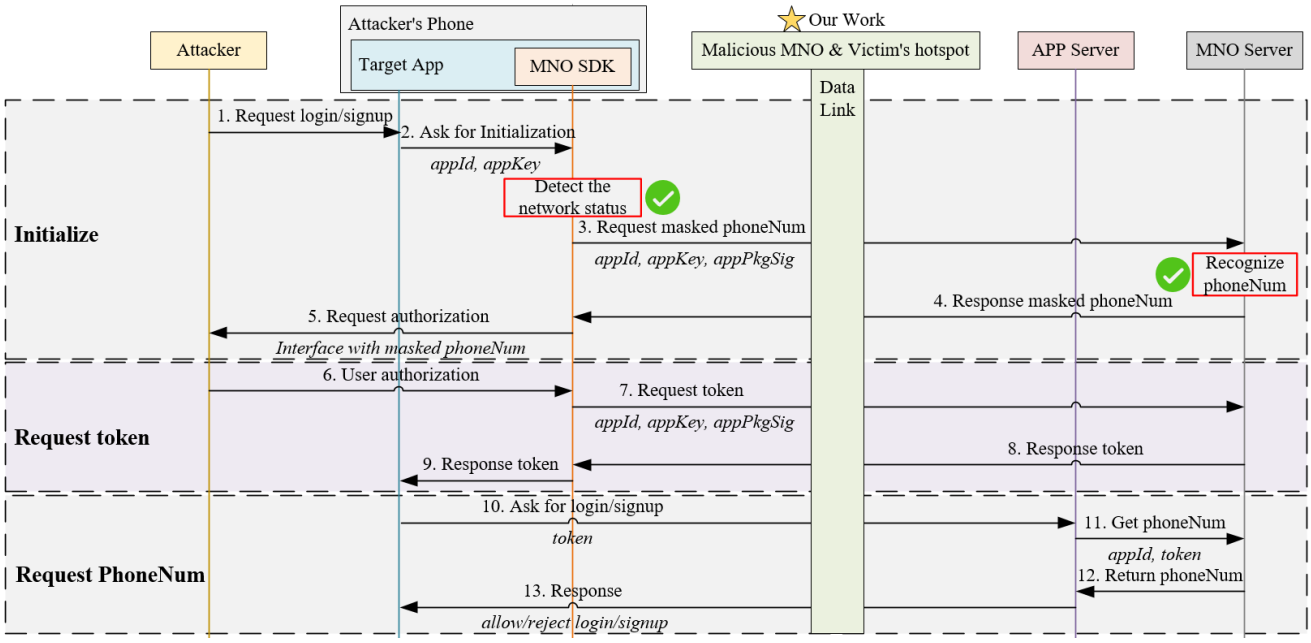

Fig. 6. The attack model exploiting the new fundamental design flaw of OTAuth services.

phone should run in a rooted environment. However, this can be easily checked by the app and thus OTAuth services are not enabled. In addition, some apps check if the function is hooked, which will also cause the method to fail. As a consequence, we need to find a way to bypass the check and we discuss it as follows.

A3. We have identified a new fundamental design flaw of OTAuth services: ***During the process of detecting the network status, OTAuth SDK only checks the parameters of the connected MNO, and cannot judge the authenticity of the currently connected MNO.*** Exploiting this vulnerability, we propose a novel method to solve the above proposed contradiction: Convert Wi-Fi signals into cellular network signals with the helper of a malicious cellular network impersonating the target MNO. Specifically, the attacker builds a malicious cellular network, whose parameters (i.e., MCC and MNC) are consistent with the MNO's parameters accessed by the victim's mobile phone, and sets the gateway of the malicious network as the victim's hotspot. At this point, the OTAuth SDK in the attacker's phone will assume that the network status is as expected.

Exploiting this novel method, the attacker can bypass the security mechanism of OTAuth services to log into the victim's app account and pose threats to privacy-sensitive data. If the victim's phone number has not been used to register any account, the attacker can register new accounts in advance on behalf of the victim. Fig. 6 shows our attack model. In an OTAuth process, the data transmitted and received by the attacker's phone will go through the data link established by the malicious MNO and the victim's hotspot. The two security procedures involving the cellular network can be perfectly bypassed. And most importantly, the entire attack process will not be considered as abnormal behaviors by the app client, app server, and MNO server. The details of the attack model will be presented in the next section.

V. An Attack Paradigm for OTAuth Services

In this section, we first present the assumptions in the attack process. Then, our attack approach is discussed which is also compared with existing attack models. Finally, we give an example of the attack effect in real life.

A. Assumptions

In our attack, the victims are subscribers of three MNOs in China. The attacker only needs to access the hotspot established through the cellular network of the victim's phone to perform the attack. For this assumption, the previous work [40] shows that an attacker can steal user data through the victim's hotspot. The results of our informal survey indicate that such an assumption is most likely to occur in the following scenarios: (1) The attacker is someone familiar to the victim. (2) The victim's hotspot password is too simple to be easily cracked. And a novel approach for Wi-Fi password recovery makes our assumption stronger [41]. And the attacker should determine the MNO to which the victim's phone number belongs. This can be done by querying the IP information of the device that accesses the victim's hotspot [42].

B. Our Attack Approach

Fig. 7 shows the scenario of our proposed attack. The most important work for the attacker is the construction of the malicious cellular network. The 4G and 5G networks provide cellular network channels for OTAuth services. The difference between 4G and 5G architectures does not affect the communication gateway access technique, which enables OTAuth services. Therefore, OTAuth services in 4G and 5G networks are almost identical. Our attack applies to both 4G and 5G networks. Even when the victim is connected to the 5G network, the attacker can successfully launch our attack by building a malicious 4G network. We choose the 4G network

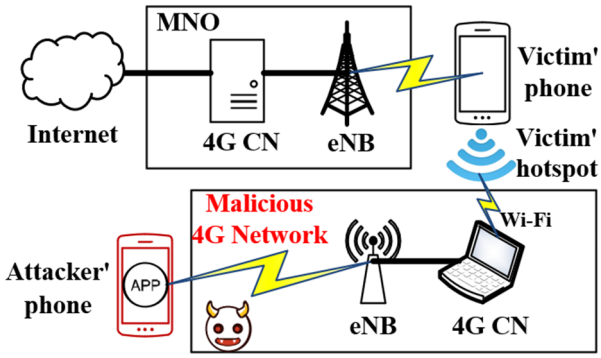

Fig. 7. The framework of our proposed attack.

and assume the victim as a user of China Unicom to illustrate the attack procedure. The attack steps for China Mobile and China Telecom users are similar. The attack procedure can be divided into the following steps:

(1) The victim uses a legitimate USIM card to access the 4G network of China Unicom and enables the mobile data. Then the victim shares the cellular network in the form of the Wi-Fi hotspot.

(2) The attacker builds a malicious 4G network to simulate China Unicom. Specifically, the PLMN of the malicious 4G network is set to 46001, and the APN is set to 3gnet to be consistent with China Unicom. This avoids the subsequent operation of configuring the APN in the mobile phone [43]. In addition, add a UE configuration information (including the IMSI, authentication key, etc.) to the UE database of the malicious network, ensuring that the first 5 bits of the IMSI are 46001. It is worth noting that this IMSI does not have to be consistent with the victim's IMSI. The DL_EARFCN of the eNB in the malicious network is set to 1650, which is in the DL_EARFCN list of China Unicom. Finally, the attacker connects the computer running the CN of the malicious 4G network to the victim's hotspot by Wi-Fi. Meanwhile, the attacker configures the gateway of the malicious CN as the network interface connected to the victim's hotspot. In this way, the attacker's phone can indirectly connect to the victim's hotspot and then access the Internet. Compared with the direct connection between the attacker's phone and the victim's hotspot, our indirect connection manner can bypass the network status check by OTAuth SDK.

(3) The attacker writes a programmable USIM card whose parameters are consistent with the UE information configured in step 2. Then the attacker inserts it into the attacker's mobile phone, and accesses the established malicious 4G network. If the APN is not set to 3gnet in step 2, the APN needs to be added to the attacker's mobile phone [43].

(4) Install an app that supports the OTAuth service on the attacker's phone. The attacker enters the app and clicks the login button. Then the attacker can imitate the victim to perform malicious actions on the app.

C. Comparison With Existing Attack Approaches

Our proposed attack approach can simultaneously address these limitations of existing attacks. Our attack is simpler, more feasible, and affects more apps.

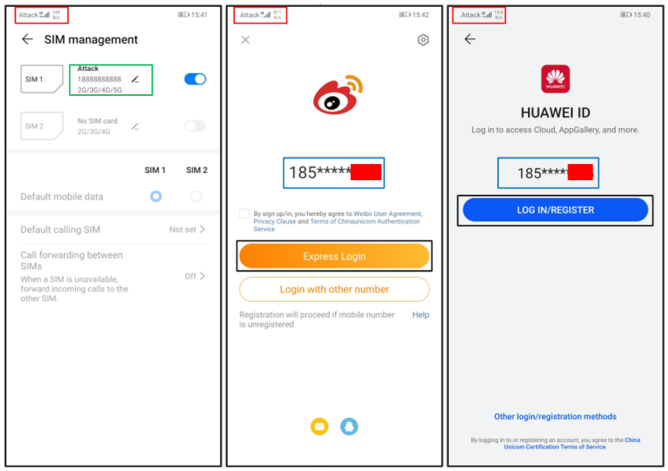

Fig. 8. Examples of the attack effect.

First, the attacker does not have to install a malicious app on the victim's mobile phone. This makes our attack more feasible. And our attack requires no extra effort for different target apps. Second, the attacker's mobile phone runs normally without being rooted. And the attacker needs not use the hook technique to bypass the check of cellular network status performed by the OTAuth SDK. These will help prevent our attack approach from being checked out. Therefore, our attack affects more apps. And our measurement results, provided in the next section, indeed confirm this.

D. An Example of the Attack Effect

We now give an example of the attack effect shown in Fig. 8, which contains three screenshots of the attacker's mobile phone. In this attack, the victim is a China Unicom subscriber with the phone number 185*****XXX. For privacy reasons, we have replaced some numbers with X. The malicious 4G network established by the attacker is named Attack marked by red boxes in Fig. 8. The phone number of the programmable USIM card in the attacker's mobile phone is 18888888888 (marked by the green box). When the attacker logs into the app supporting OTAuth services, the app will pop up the authorization interface, which displays the victim's mobile number (marked by the blue boxes). Then the attacker can click the button (marked by the black boxes) to register or log in to the app as the victim. Fig. 8 shows two apps, namely Sina Weibo (the most popular social media platform in China) and Huawei Music. Huawei Music and other Huawei official apps (e.g., Huawei Health) are logged in by Huawei ID, which can be registered with a phone number through the OTAuth service or the SMS-based authentication. In fact, these Huawei official apps do not directly embed the OTAuth SDK and on the contrary they employ the OTAuth SDK in an indirect manner. The OTAuth service of these apps is provided by the Huawei Mobile Services (HMS) Core in which the OTAuth SDK is embedded. The HarmonyOS installs HMS Core by default. Users can manually install HMS Core on the Android system. And there are no iOS versions of these Huawei official apps and HMS Core. It is worth noting that Huawei Music is not vulnerable to the attacks in the previous work [6]. However, in our work, the attacker can impersonate the victim to register

Fig. 9. The hardware used in the experiment.

a Huawei ID to log in to these apps if the phone number is not registered with Huawei ID in advance. Specifically, after accessing the malicious 4G network that sets the victim's cellular hotspot as the gateway, the attacker's phone with HMS Core installed can interact with the remote server through the victim's cellular network channel. After clicking to log in to these Huawei official apps, these apps will automatically call the OTAuth service provided by HMS Core. Therefore, the attacker can bypass the OTAuth and use the victim's phone number to register a Huawei ID to log in to these apps.

VI. Performance Evaluation of Our Attack Paradigm

To evaluate the impact of our proposed attack on real-world apps, we conduct a measurement over some popular Android, iOS, and HarmonyOS apps. Our results show that our attack can be successfully implemented against almost all apps that support OTAuth services.

A. Experimental Setup

To verify the feasibility of our proposed attack, we need to test OTAuth services of China Mobile, China Unicom, and China Telecom. We introduce our experimental setup from the following aspects, namely hardware, dataset, and measurement approach.

1) Hardware: Fig. 9 shows the hardware required for our experiment, including one computer, a card reader, four USIM cards, a radio transceiver, and three commercial mobile phones.

The operating system of the computer is 64-bit Ubuntu 16.04 LTS. The computer is equipped with srsRAN [44], which is an excellent open-source 4G software radio suite. Combined with the USRP B210, a Software-Defined Radio equipment, the srsRAN can be used to build a malicious 4G network. Detailed building tutorials are posted online (https://github.com/cellularsecurity/Build-4G-and-5G). The cost of building a malicious 4G network is about $2,200 excluding the mobile phone and the computer. To make the network more stable, the attacker can select the commercial 4G/5G simulation system Amarisoft [45]. The four USIM cards include one programmable USIM card [46] and three legal USIM cards issued by the three MNOs. The three test phones have different operating systems, namely Huawei Mate30 5G (Android), iPhone12 (iOS), and Honor V30 (HarmonyOS). When testing OTAuth services of a mobile phone with an operating system, any one of the other two mobile phones can be selected as the victim to turn on the hotspot.

To prevent our attack from affecting other normal users, we put the attacker's mobile phone and the antennas of the simulated network in the shielding box. At the same time, this also prevents the victim's mobile phone from accessing the malicious network.

2) Dataset: Our dataset includes 100 Android apps, 90 iOS apps, and 100 HarmonyOS apps. These apps are the latest version as of May 24, 2022. To better compare with previous work, our dataset is built on the work [6]. Zhou et al. [6] selected more than 1,000 apps with over 100 million downloads based on the app categories provided by Huawei App Store [47] and download statistics collected from Qimai Data [48]. We choose the top 100 of these apps as our test targets. In addition, we replace two of the apps that were banned for illegally collecting personal information with Taobao and QQ Reader. Note that since not all of these 100 apps have corresponding iOS versions, such as Huawei Music, we only test 90 iOS apps. We believe that this dataset has been able to adequately cover popular apps that users use on a daily basis. We have installed these apps on the three mobile phones with different operating systems.

3) Measurement Approach: Zhou et al. [6] proposed a mix of static and dynamic analysis methods to find those apps that are vulnerable to their proposed attack. Specifically, they focus on whether the OTAuth SDK signatures exist in the decompiled app code. If the result is true, the app is suspected of being affected by their attacks. However, such methods cannot ensure the reported suspicious apps are indeed vulnerable to their attacks. Most apps that support the OTAuth service have security features to determine if they have been hooked, which will determine the success or failure of their attacks. This will cause such methods to misjudge that these apps are also affected. In addition, the methods might assume that an app having integrated the app packing technique is immune to the attack. But it's not actually the case.

Based on the previous excellent and inspiring work [6], we choose the manual testing scheme after considering the efficiency of automated the method and the scale of the dataset. We have tested 100/90/100 Android/iOS/HarmonyOS apps for 3 MNOs' OTAuth services. In total, we conducted close to 1,000 attacks. We login or sign up an app on the victim's mobile phone normally to determine whether the app supports the OTAuth service. And we consider an app is vulnerable to our proposed attack if we actually perform malicious actions on the app as the victim on the attacker's phone.

B. Analysis of Experiment Results

Table I has summarized our measurement results. In the table, "CM", "CU", and "CT" represent China Mobile, China Unicom, and China Telecom respectively. In total, among the 100/90/100 Android/iOS/HarmonyOS apps, we have identified 73/60/72 Android/iOS/HarmonyOS apps supporting the

TABLE I
OVERVIEW OF OUR MEASUREMENT RESULTS

APP	Total	Total Affected	Result					
			MNO	Support OTAuth	Affected APP	Additional Auth.	Enter PhoneNum	Check IP
Android	100	58	CM	58	39	4	0	15
			CU	73	57	16	0	0
			CT	71	55	16	0	0
iOS	90	56	CM	57	13	3	1	40
			CU	60	54	5	1	0
			CT	59	54	5	0	0
HarmonyOS	100	57	CM	58	39	4	0	15
			CU	72	56	16	0	0
			CT	71	55	16	0	0

TABLE II
DETAILED TEST RESULTS FOR SOME POPULAR APPS

APP	China Mobile			China Unicom			China Telecom		
	Android	iOS	HarmonyOS	Android	iOS	HarmonyOS	Android	iOS	HarmonyOS
Pinduoduo	*	*	*	*	*	*	*	*	*
TikTok	√	×	√	√	√	√	√	√	√
Taobao	√	√*	√	√	√*	√	√	√	√
Kuaishou	√	√	√	√	√	√	√	√	√
Baidu	√	√	√	√	√	√	√	√	√
Sina Weibo	×	×	×	√	√	√	√	√	√
Toutiao	√	√	√	√	√	√	√	√	√
Iqiyi	√	×	√	√	√	√	√	√	√
Meituan	√	×	√	√	√	√	√	√	√
Dianping	√	×	√	√	√	√	√	√	√
Youku	√	×	√	√	√	√	√	√	√
Tencent News	√	×	√	√	√	√	√	√	√
Gaode Map	√	√	√	√	√	√	√	√	√
Ctrip	√	√	√	√	√	√	√	√	√
Meitu	√	×	√	√	√	√	√	√	√
Huawei Music	-	-	-	*	-	*	*	-	*
WPS Office	×	×	×	√	√	√	√	√	√
DingTalk	*	*	*	*	*	*	*	*	*
Moji Weather	√	×	√	√	√	√	√	√	√
Sina News	×	×	×	√	√	√	√	√	√

OTAuth services. Some apps do not support OTAuth services provided by the three MNOs at the same time, which is the reason for the difference in the number of apps supporting OTAuth services shown in Table I. We count the number of apps affected by our attack under the attack model where the attacker only shares the victim's hotspot and the victim has registered the target app with the phone number. Among these apps, 58/56/57 Android/iOS/HarmonyOS apps in total are affected. Table II shows the detailed test results of 20 popular apps. In the table, "√" indicates that the app can be attacked. "√*" indicates that the attacker should know the victim's phone number. "*" indicates that the attacker can register an account with the victim's mobile phone number. "×" indicates that the attacker needs to obtain the victim's IP assigned by the MNO to launch the attack. "-" indicates that the app does not support the OTAuth service provided by the specific MNO or lacks the iOS version.

1) Discussion About Apps: Table I shows that our attack is not applicable to some apps that support OTAuth services mainly due to the following reasons: (1) Adopt additional verification for new devices. For example, when the attacker tries to log in to the victim's account on a new device, Pinduoduo will require an additional SMS-based authentication. However, if the victim has not registered for these apps, the attacker can perform unauthorized registration in advance. Among these apps that support the OTAuth services, 16 apps in total adopt additional verification for new devices. And the detailed test results for the three operating systems and the three MNOs are somewhat different as shown in Table I. (2) Need to enter the phone number. Before using OTAuth services, the app requires the user to enter a full phone number. This requirement is only for China Mobile and China Unicom users of Taobao under iOS. However, attackers can log in to other apps in advance to obtain the full phone number (see Section VIII). (3) Check the local IP. Only China Mobile's OTAuth SDK will check the local IP. If the attacker does not know the victim's IP assigned by China Mobile's cellular network, the attacker's IP assigned by the malicious network may be different from the victim's IP. This makes our attack no longer applicable to apps that are equipped with the SDK. Although the attacker can obtain the target IP through social engineering or installing malware on the victim's phone, this greatly increases the cost of the attack.

2) Discussion About MNOs: From the test results shown in Table II, it can be concluded that OTAuth services provided by the 3 MNOs are slightly different. Since the SDKs provided by China Unicom and China Telecom do not check the local IP, our attack poses a greater threat to users of these two MNOs.

3) Discussion About Operating Systems: The test results of OTAuth services provided by an MNO for the three operating systems may be different. For example, our attack on the TikTok account of a China Mobile user can be carried out on the Android and the HarmonyOS, but it fails on the iOS. However, considering the interoperability of accounts on different operating systems, our attack only needs to apply to one of the three operating systems for the attacker to log into the victim's account.

4) Further Impact of Our Attack: We find that some app accounts are linked to each other. For example, the user can manually choose to log in to the UC Browser account with the Sina Weibo account. This inspires us that an attacker can first exploit our attack to log in to an app that supports OTAuth services, and then use the app account to log in to other apps. In this way, the number of apps affected by our attack is further increased.

5) Comparison With Previous Work: Compared to the tests conducted by Zhou et al. [6] for these 100 apps, our tests are specific to different MNOs and take into account the possible variability in the implementation of OTAuth services provided by these MNOs. It turns out that these MNOs do have different implementations. In addition, among these 100 apps, our attack poses security threats to a larger number of apps than the attack proposed by in [6], such as Ctrip, Meituan, Huawei Music, etc. Considering the different versions of these apps, we then replaced these apps using Wandoujia [49] with versions consistent with those tested in [6]. The results show that our attack still works for some apps. It is worth mentioning that we found that some apps (e.g., Gaode Map) that support OTAuth services and do not have additional authentication are also not necessarily affected by the attacks proposed in [6]. These apps check whether the phone is rooted, which is an important prerequisite for the attack in [6]. If the phone is indeed running in a rooted environment, these apps disable OTAuth services. In our attack, the attacker's phone does not need to be rooted, so these security enhancements do not affect our attack.

VII. A SURVEY ABOUT THE ATTACK EFFECT AND INFLUENCE

Recently, nearly 600 apps support OTAuth services and we believe that in the future, more apps will employ the OTAuth

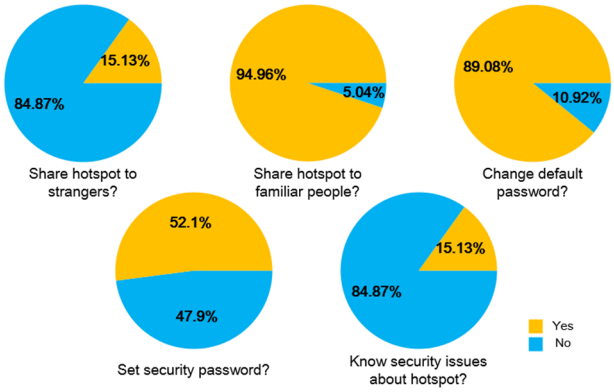

Fig. 10. Hotspot related survey results.

service. This can be explained by the fact OTAuth is quite convenient for the users compared with other authentication schemes. In this case, considering the strong correlations between app accounts, our attack affects a large number of apps. We have conducted an informal survey to evaluate the feasibility of the precondition for our proposed attack (i.e., connecting to the hotspot established by the victim's cellular network). We received responses from 119 volunteers, and Fig. 10 shows the survey results. This result indicates that our proposed attack model has certain feasibility in real life.

Specifically, among all the participants, only 15.13% of the users are willing to share the hotspot to strangers, but 94.96% of the users agree to open the hotspot for familiar people. This inspires us that the attack is most likely to occur in the following scenario: The attacker is a colleague of the victim, and the attacker attempts to login into the victim's app account to steal privacy. In addition, 10.92% of the users will not change the default hotspot password, which is often relatively simple. Among the volunteers who have changed their passwords, 47.90% of the users do not set a password with a higher security level (including letters, numbers, symbols, and more than 8 characters). It is possible for an attacker to brute force the password to access the victim's hotspot. Lastly, we assess users' sensitivity to the hotspot security. Only 15.13% of the users know some security issues related to the phone hotspot. But these security issues have nothing to do with OTAuth services. This fact facilitates the implementation of our proposed attack.

VIII. Other Related Implementation Weaknesses of OTAuth Services

Some additional implementation flaws of OTAuth services are discussed in existing work, such as insecure token usage (including token reuse and too long token validity period), authorization without user consent, and plain-text storage of sensitive information. Our analysis shows that these insecure implementations have been fixed. However, we have identified some other implementation weaknesses involving both the OTAuth SDK and apps when performing measurements over a set of apps.

A. Insecure Implementation of OTAuth SDK

Our measurements show that China Unicom's and China Telecom's SDKs do not verify the local IP address of mobile phones, while all three MNOs do not check the type of cellular network. That is to say, when the victim accesses the MNO's 5G (resp. 4G) network, our proposed attack can be also successfully launched through building a malicious 4G (resp. 5G) network. This prevents the victim's phone from accessing the malicious network set up by the attacker. Therefore, this implementation flaw is conducive to the initiation of our attack.

B. Insecure Implementation of the App Client

We found that two insecure implementations of several apps can lead to the leakage of phone numbers. First, some app clients directly display the plain-text phone number in the app's user-profile interface. This indicates that the app server returns the phone number in plain-text to the app client. An attacker may hijack the communication between the app client and the app server and obtain the plain-text phone number. Second, we found that attackers may infer the plain-text phone number through the masked phone number. As shown in Fig. 1, the masked phone number is displayed to the user in the authorization interface pulled up by the app. And the masked digits set by these apps are different. For example, Alipay sets 6 digits, while Kuaishou only sets 4 digits. If the masked digits are too few, an attacker may infer the complete mobile phone number based on the corresponding IMSI number, which can be obtained through the IMSI-catcher [50], thereby leading to the disclosure of the user's sensitive information. In Fig. 1, Kuaishou shows a masked phone number (185****8XXX) of a China Unicom user. The IMSI of the user's USIM card is 460018771XXXXXX. Then we can infer that the user's phone number is 18518778XXX. In addition, after an attacker logs in to some app accounts with a victim's phone number, these apps will not remind the victim that these accounts have been logged in elsewhere even if these accounts are already logged in on the victim's phone.

IX. Countermeasures

We first analyze the shortcomings of the security strategies proposed in previous work and then propose our countermeasures to effectively enhance the security of OTAuth services.

We summarize the defenses in [6] as follows: (1) Adding user-input data into the login request. OTAuth services could require users to input some additional information, such as the phone number, etc. However, this may affect the user experience. Moreover, an attacker could collect the necessary information in advance. (2) Adding OS-level support. This requires OS to provide support to distinguish tokens from different apps. However, since this requires deep cooperation between MNOs and OS vendors, the strategy is not feasible. Most importantly, this countermeasure is not effective against our proposed attack. Note that our attack will not be noticed by the OS, apps, or the OTAuth SDK.

We argue that the above defenses are ineffective. In this paper, we propose the following countermeasures involving both OTAuth SDK developers and app developers.

A. For OTAuth SDK Developers

The SDK should obtain more parameters without violating the premise of collecting personal information and submit

them to MNO's authentication server for inspection. For example, in addition to existing parameters, add checks for the cellular network type (4G or 5G) and the local IP address. These security enhancements will greatly increase the implementation cost of our proposed attack. The attacker needs not only the victim's hotspot but also the victim's local IP address. Note that this can also invalidate the attack through the victim's hotspot [6]. In addition, due to checking the cellular network type, the malicious network must be consistent with the cellular network to which the victim is connected. Then the malicious network may interfere with the victim's signal and cause the attack to fail. The attacker must perform additional actions to solve this problem, which will make the attack more difficult.

B. For App Developers

The app developers can carry out security enhancements to apps from the following points: (1) Adopt additional authentication factors for new devices. When using OTAuth services, check whether the account is logged in on a new device. If the statement is true, the app should require the additional verification, such as SMS-based authentication, entering the last four digits of the user's ID card number, etc. (2) Check whether the mobile phone is rooted. The OTAuth service should be disabled in a rooted runtime environment. This can effectively resist the two attacks mentioned in [6]. In addition, the hook detection method [51] can also be adopted so that when attackers attempt to hook functions related to OTAuth, the service can be disabled in time. (3) Check the app client version. If the app client is not the latest version, OTAuth services should be prohibited to prevent attackers from using older versions of the app client without these security enhancements. (4) Set login reminder. When an account is logged in on the other device, the app server can send a reminder message containing the information of the device to the user.

X. Conclusion

In this paper, we analyze the cellular network based OTAuth service in detail, which is adopted by various mobile apps. We mainly focus on the cellular network that plays an important role in this authentication service and identify a new fundamental design flaw. Based on this flaw, we design an attack, which can be exploited to bypass the OTAuth service and log into the victim's app account. Compared with existing approaches, our scheme is more mature. Our attack can be implemented only using the victim's hotspot, without the app being hooked or running in a rooted environment. This enables our attack to be undetectable by the app or the OTAuth SDK. In addition, we have conducted a large-scale measurement over a set of apps (including Android, iOS, and HarmonyOS apps) to further evaluate the impact of the attack against OTAuth services provided by three MNOs in China. The results show that almost all apps with OTAuth services are affected by our attack. We also found some other implementation flaws in our testing. Given that the security enhancements proposed in previous research cannot effectively protect against our attack, we finally discussed some countermeasures from the perspectives of both OTAuth SDK developers and app developers to mitigate these threats.

References

[1] S. Holtmanns and I. Oliver, "SMS and one-time-password interception in LTE networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[2] Y. Zheng, L. Huang, H. Shan, J. Li, Q. Yang, and W. Xu, "Ghost telephonist impersonates you: Vulnerability in 4G LTE CS fallback," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Las Vegas, NV, USA, Oct. 2017, pp. 1–9.

[3] K. Nohl and S. Munaut, "GSM sniffing," in *Proc. 27th Chaos Commun. Congr.*, Berlin, Germany, 2010. [Online]. Available: https://www.maxrev.de/files/2012/04/1783_10122827c3gsm_sniffingnohl_munaut.pdf

[4] GSM Association. (2022). *Connecting Through a Secure Digital Identity With Mobile Connect*. [Online]. Available: https://www.gsma.com/identity/mobile-connect

[5] GeekPwn. (2019). *Hall of Fame*. [Online]. Available: http://hof.geekpwn.org/zh/index.html

[6] Z. Zhou, X. Han, Z. Chen, Y. Nan, J. Li, and D. Gu, "SIMulation: Demystifying (insecure) cellular network based one-tap authentication services," in *Proc. 52nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Baltimore, MD, USA, Jun. 2022, pp. 534–546.

[7] Oleavr. (2022). *Frida: A World-Class Dynamic Instrumentation Framework*. [Online]. Available: https://frida.re/docs/functions/

[8] Wikipedia. (2022). *List of Mobile Network Operators*. [Online]. Available: https://en.wikipedia.org/wiki/List_of_mobile_network_operators

[9] Z. Zhang, Y. Song, and Y. Xia, "Using short message service (SMS) to deploy Android exploits," in *Proc. 5th Int. Conf. Multimedia Inf. Netw. Secur.*, 2013, pp. 890–893.

[10] F-Secure. (2022). *Trojan: Android/Crusewind*. [Online]. Available: https://www.f-secure.com/v-descs/trojan_android_crusewind.shtml

[11] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, "Exploiting open functionality in SMS-capable cellular networks," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2005, pp. 393–404.

[12] J. Liu, Y. Yu, and F. X. Standaert, "Small tweaks do not help: Differential power analysis of MILENAGE implementations in 3G/4G USIM cards," in *Proc. Eur. Symp. Res. Comput. Secur.*, Cham, Switzerland, 2015, pp. 468–480.

[13] K. Lee, B. Kaiser, and J. Mayer, "An empirical study of wireless carrier authentication for SIM swaps," in *Proc. 16th Symp. Usable Privacy Secur.*, 2020, pp. 61–79.

[14] Y. Zhang et al., "Lies in the air: Characterizing fake-base-station spam ecosystem in China," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2020, pp. 521–534.

[15] L. Huang, "Forcing targeted LTE cellphone into unsafe network," in *Proc. HITB AMS Secur. Conf.*, 2016. [Online]. Available: https://archive.conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/D1T1-Lin-Huang-Forcing-a-Targeted-LTE-Cellphone-into-an-Eavesdropping-Network.pdf

[16] Android. (2022). *Android 12 Adds an Option to Disable 2G Modem on Phones That Ship With it*. [Online]. Available: https://www.xda-developers.com/android-12-disable-2g-modem/

[17] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu, "New security threats caused by IMS-based SMS service in 4G LTE networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 1118–1130.

[18] Z. Cui, B. Cui, J. Fu, and R. Dong, "Security threats to voice services in 5G standalone networks," *Secur. Commun. Netw.*, vol. 2022, pp. 1–13, Sep. 2022.

[19] (2022). *Harassing Calls*. [Online]. Available: http://315.cctv.com/

[20] H. Wang et al., "Vulnerability assessment of OAuth implementations in Android applications," in *Proc. 31st Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, Dec. 2015, pp. 61–70.

[21] H. Wang, Y. Zhang, J. Li, and D. Gu, "The Achilles heel of OAuth: A multi-platform study of OAuth-based authentication," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, New York, NY, USA, Dec. 2016, pp. 167–176.

[22] S. W. Park and J. H. Yi, "Multiple device login attacks and countermeasures of mobile VoIP apps on Android," *J. Internet Serv. Inf. Secur.*, vol. 4, no. 4, pp. 115–126, 2014.

[23] W. Song et al., "App's auto-login function security testing via Android OS-level virtualization," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, May 2021, pp. 1683–1694.

[24] Defcon 25 Hacker Conference. (2017). *Bypassing Android Password Manager Apps Without Root*. [Online]. Available: https://teamsik.org/wp-content/uploads/2017/10/DEFCON25-pw-manager.pdf
[25] A. Bianchi, E. Gustafson, Y. Fratantonio, C. Kruegel, and G. Vigna, "Exploitation and mitigation of authentication schemes based on device-public information," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, Dec. 2017, pp. 16–27.
[26] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking LTE on layer two," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 1121–1136.
[27] Q. Wang and D. Wang, "Understanding failures in security proofs of multi-factor authentication for mobile devices," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 597–612, 2023.
[28] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Advances in Cryptology—EUROCRYPT 2018* (Lecture Notes in Computer Science), vol. 10822. Tel Aviv, Israel: Springer, 2018, pp. 456–486.
[29] C. Wang, D. Wang, Y. Duan, and X. Tao, "Secure and lightweight user authentication scheme for cloud-assisted Internet of Things," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2961–2976, 2023.
[30] J. D. Clercq, "Single sign-on architectures," in *Proc. Int. Conf. Infrastruct. Secur.*, 2002, pp. 40–58.
[31] Technical Specification Group Service and System Aspects; 3GPP System Architecture Evolution (SAE); Security architecture (Rel 17), Standard 3GPP TS 33.401 V17.3.0, 3rd Generation Partnership Project, Sep. 2022.
[32] *Technical Specification Group Services and System Aspects; 3GPP Security Architecture and Procedures for 5G System (Rel 17)*, 3GPP, document TS 33.501 V17.7.0, Sep. 2022.
[33] *Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) Protocol for Evolved Packet System (EPS); Stage 3 (Rel 17)*, 3GPP, document TS 24.301 V17.8.0, Sep. 2022.
[34] *Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) Protocol for 5G System (5GS); Stage 3 (Rel 17)*, 3GPP, document TS 24.501 V17.8.0, Sep. 2022.
[35] China Mobile. (2022). *China Mobile Capability Store*. [Online]. Available: https://open.10086.cn/#/capability/14
[36] Shenzhen Hexun Huagu Information Technology Co., Ltd. (2022). *Aurora Mobile Security Verification*. [Online]. Available: https://www.jiguang.cn/en/identify
[37] Shanghai Blue Culture Communication Co., Ltd. (2022). *Chuanglan Shanyan*. [Online]. Available: https://shanyan.253.com/
[38] Google Developers. (2022). *Sign Your App*. [Online]. Available: https://developer.android.com/studio/publish/app-signing
[39] Apple. (2022). *Apple Platform Security*. [Online]. Available: https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf
[40] A. Coletta, G. Maselli, M. Piva, D. Silvestri, and F. Restuccia, "My SIM is leaking my data: Exposing self-login privacy breaches in smartphones," 2020, *arXiv:2003.08458*.
[41] M. Sebastian, P. Vinod, and S. Vishnudev, "Presenting a novel method for Wifi password recovery," in *Proc. Nat. Conf. Emerging Comput. App.*, 2020, pp. 1–3.
[42] Xiamen Manyou Science and Technology Co., Ltd. (2022). *IP Query*. [Online]. Available: https://www.ip138.com/
[43] srsRAN. (2022). *Connecting the COTS UE*. [Online]. Available: https://docs.srsran.com/en/latest/app_notes/source/cots_ue/source/index.html
[44] srsRAN. (2022). *Your Own Mobile Network*. [Online]. Available: https://www.srslte.com/
[45] Amarisoft. (2022). *AMARI Callbox Series*. [Online]. Available: https://www.amarisoft.com/products/test-measurements/amari-lte-callbox/
[46] Sysmocom. (2022). *Sysmocom SIM/USIM/ISIM Cards*. [Online]. Available: https://www.sysmocom.de/products/sim/sysmousim/index.html
[47] Huawei App Store. (2022). *Huawei App Store*. [Online]. Available: https://appstore.huawei.com/
[48] Beijing Qimai Technology Co., Ltd. (2022). *Qimai Data (Formerly ASO100): Professional Mobile Product Business Analysis Platform*. [Online]. Available: https://www.qimai.cn/
[49] Wandoujia. (2022). *The Apps Provided by the Third-Party Markets*. [Online]. Available: https://www.wandoujia.com/
[50] S. F. Mjølsnes and R. F. Olimid, "Easy 4G/LTE IMSI catchers for non-programmers," in *Proc. Int. Conf. Math. Methods, Models, Archit. Comput. Netw. Secur.*, Cham, Switzerland, 2017, pp. 235–246.
[51] M. Szczepanik, I. J. Jóźwiak, and P. P. Jóźwiak, "Android hook detection based on machine learning and dynamic analysis," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Cham, Switzerland, 2020, pp. 1321–1329.

Zhiwei Cui received the B.E. degree from the School of information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), in 2019, where he is currently pursuing the Ph.D. degree with the School of Cyberspace Security. His research field focuses on cellular network security.

Baojiang Cui received the B.S. degree from the Hebei University of Technology, China, in 1994, the M.S. degree from the Harbin Institute of Technology, China, in 1998, and the Ph.D. degree in control theory and control engineering from Naikai University, China, in 2004. He is currently a Professor at the School of Computer Science, Beijing University of Posts and Telecommunications, China. His main research interests include detection of software, cloud computing, and the Internet of Things.

Junsong Fu received the Ph.D. degree in communication and information system from Beijing Jiaotong University in 2018. He is currently an Assistant Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include network security and information privacy issues in the internet, mobile networks, and the Internet of Things.

Bharat K. Bhargava (Life Fellow, IEEE) is a Professor of computer science at Purdue University. He is the Founder of the IEEE Symposium on Reliable and Distributed Systems, the IEEE Conference on Digital Library, and the ACM Conference on Information and Knowledge Management. He is the Editor-in-Chief of four journals and serves on over ten editorial boards of international journals. He has published hundreds of research articles and has won five best paper awards in addition to the Technical Achievement Award and Golden Core Award from IEEE.