When Harry Met Tinder: Security Analysis of Dating Apps on Android

Kuyju Kim, Taeyun Kim, Seungjin Lee, Soolin Kim, and Hyoungshick Kim(✉)

Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Republic of Korea
{kuyjukim,taeyun1010,jine33,soolinkim,hyoung}@skku.edu

Abstract. As the number of smartphone users has increased, so has the popularity of dating apps such as Tinder, Hinge, Grindr and Bumbler. At the same time, however, many users have growing privacy concerns about these applications disclosing their sensitive and private information to other service providers and/or strangers. This is particularly exacerbated due to the nature of dating apps requiring access to users' personal contents such as chat messages, photos, video clips and locations. In this paper, we present an analysis of security and privacy issues in popular dating apps on Android. We carefully analyze the possibility of software vulnerabilities on the five most popular dating apps on Android through network traffic analyses and reverse engineering techniques for each dating app. Our experiment results demonstrate that user credential data can be stolen from all five applications; three apps may lead to the disclosure of user profiles, and one app may lead to the disclosure of chat messages.

Keywords: Dating application · Privacy · Vulnerability · Android

1 Introduction

As the number of smartphone users has increased, online dating apps (e.g., Tinder[1], Amanda[2], Glam[3], DangYeonsi[4] and Noondate[5]) on smartphones have become increasingly popular. In 2017, for example, `Tinder`, which is one of the most popular dating apps, acquired more than 50 million users [5]. The revenue of the "online dating" industry may reach as much as 1.3 billion USD in 2018. This revenue is expected to develop at an annual growth rate of 3.9%, resulting in a market volume of 1.6 billion USD in 2022 [4].

However, people also have growing privacy concerns about disclosing their sensitive and private information to unauthorized third parties [9,11,19,22].

[1] Tinder: https://tinder.com/.
[2] Amanda: http://amanda.co.kr/.
[3] Glam: http://www.glam.am/.
[4] DangYeonsi: https://www.facebook.com/DangYeonsi.
[5] Noondate: http://noondate.com/.

N. Gruschka (Ed.): NordSec 2018, LNCS 11252, pp. 454–467, 2018.
https://doi.org/10.1007/978-3-030-03638-6_28

Because dating services generally collect and store users' personal information (e.g., chat messages, sexual preferences, ethnic identity, educational level, political views, music and food tastes, pictures, videos and user location), this information is likely personally identifiable and very sensitive to users. For example, a journalist from `The Guardian` who had been using the Tinder app found that Tinder had collected about 800 pages of information about her, including information on Facebook's "*like*" feature, Instagram's photos, her education, the men's age range she was interested in and the number of her Facebook friends [6].

There have been several previous studies that inspected privacy issues in online dating apps. For example, Farnden et al. [13] analyzed what personal information is stored in nine dating apps and how it was disclosed. Shetty et al. [22] found that man-in-the-middle (MITM) attacks could be launched and pose a serious threat of privacy breaches in several dating apps.

In this paper, we extend these previous studies by focusing on the detection of software vulnerabilities (specifically related to privacy breaches) in online dating apps on Android, and we designed a generic framework to identify them. Our key contributions are summarized as follows:

- We present a generic framework to analyze software vulnerabilities in dating apps on Android through (1) packet analysis, (2) API hooking, (3) storage analysis, and (4) code decompilation.
- To show the feasibility of the proposed framework, we analyze the top five most popular online dating apps in Android. Our experiment results demonstrate that user credential data can be stolen from all the five apps; two apps may lead to the disclosure of user profiles, and one app may lead to the disclosure of chat messages.

The rest of the paper is organized as follows. Section 2 briefly summarizes security-related features for typical dating apps. Section 3 enumerates the particular privacy issues which we target in dating apps on Android. Section 4 presents analysis methods used to identify software vulnerabilities in dating apps that can be exploited in privacy breaches. Section 5 presents our analysis results, and Sect. 6 suggests defense mechanisms to prevent the risk such privacy breaches. The related work is summarized in Sect. 7. Finally, Sect. 8 concludes with a summary of the research results and suggests our directions for future research.

2 Security-Related Features for Online Dating Apps

In this section, we briefly explain the security-related features that are required for a typical mobile dating application.

User Account Management. An online dating service is a platform in which users create an account and share personal data (e.g., user profile, messages, photos, interests and preference) with other users with the goal of finding a suitable partner. Therefore, user account management is crucial. On smartphone applications, automatic login is popularly used by storing user credentials in files and offline databases. Because smartphones' small keyboards make it difficult to type complex login information, many users are attracted to the convenience of auto-login capabilities. Unsurprisingly, user credential data is also a very attractive target for attackers who want to steal users' identities [10]. In this paper, we are motivated to investigate whether user credential data in dating apps is securely protected.

Matchmaking. In online dating services, matchmaking is useful to find potential partners. For successful matchmaking, the construction of accurate user profiles (age, location, and preferences) is key. Therefore, a user is usually asked to create his/her profile during user registration. Because sensitive user data can be collected from user profiles, we also examine the potential risk of unintended personal information being exposed in dating apps.

Communication Between an App and the Server. In general, the conventional client-server model is the most widely used method for online dating services. In an online dating service, communication channels between the application and the servers in the dating service can be potential attack vectors and should be securely protected from unauthorized access.

3 Threat Model

In this section, we describe the types of attackers that we target in this work, and we discuss which assets need particular attention to be protected against those attackers.

3.1 Attacker Types

We consider three different types of attackers: (1) Network sniffer, (2) Anonymous user, and (3) Co-located attacker. The common goal of all types of attackers is to access user personal data without proper authorization from an online dating service.

Network Sniffer. The first type of attacker is a *network sniffer*. This attacker is only capable of intercepting and modifying the packets between a dating app and the servers of the target online dating service through a traffic analysis tool.

Anonymous User. The second type of attacker is a registered user in an online dating service. We call this attacker *anonymous user*. This attacker can create an account for the service and download the client app to communicate with the dating server. These attackers are not only capable of intercepting and modifying the packets between a dating app and the servers through a traffic monitoring tool, but are also capable of analyzing the behaviors (e.g., authentication methods, encryption methods, message payload structures, etc.) of the target dating app. Additionally, they can leverage debugging tools and reverse engineering techniques in their attacks because the dating app is under the attacker's full control. Anonymous users can also install a proxy on the user's smartphone in order to intercept HTTPS traffic and decrypt it on the proxy.

Co-located Attacker. The third type of attacker is co-located with the victim and may have access any storage files on the victim's smartphone. In Android, there are four different local storage options: (1) Shared Preferences, (2) Internal Storage, (3) External Storage, and (4) SQLite Databases (see https://developer.android.com/guide/topics/data/data-storage). This type of attacker is possible in many practical situations. For example, malware on Android devices is frequently able to gain read access to the victim's files. As another example, an intelligence agency can try to gain access to a victim's data on a smartphone through debugging tools (e.g., Android Debug Bridge (ADB)) after the intelligence agency is in possession of the victim's smartphone. We call this attacker *co-located attacker*.

Finally, we assume that the victim's smartphone is already rooted or will be rooted by the co-located attacker because some storage spaces on Android (i.e., Shared Preferences) can be accessed only on rooted devices.

3.2 Assets in Dating Apps

User Profile. In general, online dating services ask users to enter their personal information to create their profiles, and they use this information to recommend potential partners to each user. Unlike conventional social network services, however, more sensitive personal information is often needed to create user profiles. For example, a user's private information such as hobbies, interests, occupations, age, religious preferences, and/or sexual orientation can be included in the user profile. Most users would assume that such service providers implement significant measures to protect user profile information from unauthorized access and mismanagement through access control policies and mechanisms. In practice, however, user profile information can often be harvested (e.g., through *enumeration* attacks [16]).

Location Information. Many dating apps collect their users' location information in order to provide location-based recommendation services. Because many users are concerned about revealing their private location information, many service providers use location-based services without the need to reveal users' specific private information. For example, the list of nearby users using the same dating app is only displayed instead of other users' exact location information. However, this information can also be misused to infer a target user's location by means of triangulation [18].

User Credential. In online dating apps, users have to create a user account to use their dating services. For users' convenience, most dating apps support the automatic login feature, which results in storing user credentials (or login credentials) as cookies or tokens on a user's mobile device. In practice, however, user credentials can often be stolen by means of *user credential cloning* attacks [10] and Cross-Site Scripting (XSS) attacks.

Chat Messages. In many cases, dating apps provide an instant messaging service for their users. That is, dating app users can *secretly* exchange text messages, voices, pictures and videos with their potential partners using the instant messaging feature. Because this communication typically includes sensitive or private information, most dating apps try to protect the stored data via encryption so that only the authorized dating app itself can access the data.

4 Analysis Methods

In this section, we present in detail the methods we used for analyzing the dating apps on Android. An overview of our analysis framework is presented in Fig. 1. Given an APK file of the target dating app, we used several tools to effectively analyze potential privacy breaches that may occur in the dating app.

To analyze dating applications, we downloaded the APK file of the target dating application from an application mirroring site and installed the downloaded APK file. Following this, we conducted the analysis in two ways: static and dynamic. To perform static analysis, we extracted the storage of the apps from the device and also decompiled the APK file to analyze the reconstructed source code. In addition, we performed dynamic analysis by using packet analysis techniques and hooking. The four analysis methods can be summarized as follows:

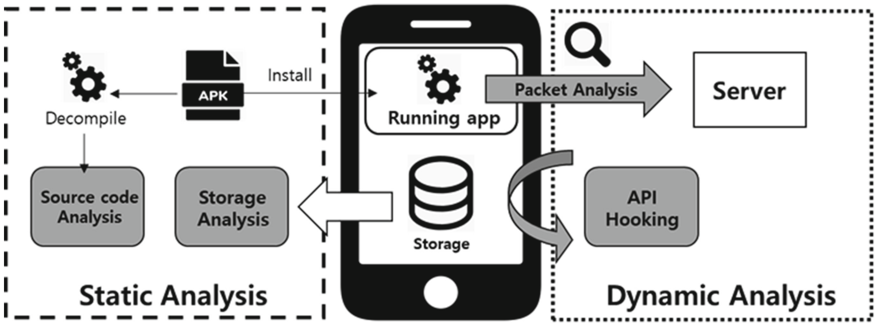


Fig. 1. Flow of privacy issue analysis in mobile applications.

Packet Analysis. Most Android applications communicate with the server when the apps are running. Therefore, we analyze the packets transmitted to the server when the apps start to communicate personal information in the packets. In an Android environment, *tpacketcapture* [7] can be used to inspect packets generated by an application. When an Android app sends packets to communicate with a web server, we can easily capture and analyze the packets via a web proxy. In this case, after configuring a proxy tool, it is possible to collect all packets for HTTP as well as HTTPS using *fiddler* [2] if a *fiddler* certificate is installed on the Android device. However, many recent apps use the SSL (Secure Socket Layer) pinning technique to prevent changing and analyzing certificates with a debugging tool like *fiddler*. The SSL pinning technique utilizes a hard-coded certificate in the APK, and when the application starts, it uses this hard-coded certificate instead of any other available certificates. To bypass the SSL pinning scheme, we used the *ssl unpinning.apk* provided by the *Xposed framework*[6]. Xposed is a framework for modules that can manipulate the flow of the system and apps without changing the APK file.

We analyze whether the IDs and passwords of users are securely delivered to the server when users attempt to login to dating applications. Most of the communication between the server and the client in dating apps take place under the HTTPS (HyperText Transfer Protocol Secure) protocol. User credentials are also sent to the server during the login process as an HTTPS request, and the server sends a cookie or a token in response. After the user logs in, post-authentication is continuously performed with the issued token or cookie to maintain the login status. However, if SSL/TLS is misused, an attacker can successfully decrypt HTTPS via MITM (Man-In-The Middle attack) attacks. Additionally, if attackers are able to bypass the HSTS (HTTP Strict Transport Security) configuration, they may be able to acquire sensitive information in plaintext form.

API Hooking. API hooking is an attack method that can be launched on an Android APK to output a specific value or change the behavior of an application by intercepting user input at a specific point without modifying the source code. At this time, repackaging the APK is not necessary. We take advantage of the application *frida* [3]. *Frida* implements hooking with javascript and uses a framework called *appmon* to monitor and automate API hooking, which greatly facilitates the entire hooking process. *appmon* provides the scripts for the basic APIs provided by Android (crypto, database, file system, etc.) and enables analysis and detection of potentially vulnerable behavior such as weak encryption, database storing, and file storing through Android API hooking.

In many mobile apps, various cryptographic techniques are used to protect sensitive data. If attackers have the ability to obtain some information such as the key or the initial vector for encryption by analyzing the app, they can easily decrypt the ciphertext acquired from the communication session with the server.

[6] http://repo.xposed.info/.

We analyze whether the cryptographic techniques are used correctly in dating apps by using API hooking.

Storage Analysis. After installing the package, the data including shared preferences, databases, and files are stored in a predetermined path for each package: /data/data/[package name]. An application stores most of the information that it uses in its path in various forms. It stores important information that the app needs to work, such as configuration files, user credentials, personal information, etc.

Therefore, we analyze the information stored in the path to check whether any personal data is stored.

Code Decompilation. To understand the low-level behavior of the application, we decompile the APK file and inspect the resulting Java code to analyze the storage, encryption, hashing, and obfuscation techniques used on any personal data. Through a source code analysis, an attacker can typically find more complex vulnerabilities that could not be found simply by running the application. Furthermore, he may be able to invoke unauthorized services or activities that would never be invoked when the app is running. We also repackage the APK after changing *smali* code to check the storage and processing of the personal data.

5 Experiments

To conduct our experiments, we investigated the security of the four assets in dating apps mentioned in Sect. 3.2. We selected 5 applications from the top 10 applications in Korea and in the US Google Play Store listed in `appannie` [1] as of March 25th, and we analyzed the attack feasibility under the threat models mentioned in Sect. 3.1. The five applications we chose are Tinder, Amanda, Noondate, Glam and DangYeonsi. In our experiments with these applications, we used a rooted Google Nexus 5 running Android 8.0. We also used a PC for manipulating packets through a Fiddler proxy server and for API hooking.

5.1 Network Sniffer

To analyze and sniff network traffic, we installed a Fiddler certificate on the mobile device, and we performed an analysis of the captured packets after setting up an HTTP proxy server on the PC. In this case, if an attacker successfully decrypts HTTPS via MITM, the data can be exposed. However, MITM attacks were unsuccessful because all five apps implemented properly configured SSL/TLS.

5.2 Anonymous User

User Profile. In dating apps, users often have to use points or cash before they can view profiles of other users who interest them. However, if profiles are requested continuously, users' profiles can be collected, and queries can be made without using any cash or points. In dating apps, every profile is given a profile index. In the case of profile inquiry, if the profile index is the result of any hash function, it is difficult for an attacker to request the profile of a certain user, or even for the profile of any user because he first has to know the hash value before he can query the profile.

An HTTP proxy tool like Fiddler can collect packets and manipulate collected packets. In order to collect user profile information, we first tested if such information can be collected by changing a parameter called profile indexes in the profile inquiry packet. As shown in Table 1, of the five apps we analyzed, three apps used consecutive numbers as profile indexes, and the remaining two used random numbers as profile indexes. For the apps that used consecutive numbers, except for the N app, user profiles could be viewed by the attacker. However, for the apps that used random numbers, user profiles could not be viewed.

In dating apps, personal information such as the user's email information is not visible in the application, but there are cases where it is actually visible in the packet. In one of the three apps where profiles could be stolen, we extracted emails during profile collection by manipulating and sending the profile request packet. A total of 883 email addresses were extracted from 1,000 different profiles, and it took 121.81 seconds. Multiple trials were performed to extract email addresses, and we confirmed that similar results could be repeatedly obtained since the server did not block incoming profile request packets. All email addresses obtained were deleted after the experiments.

Location Information. We also analyzed the user's location exposure in dating apps. Of the five apps, four were collecting location information, and among these four apps that collect location information, three were exposing distance information to users. These three apps send GPS location information to the server. When the attacker requests nearby user information, the server response contains the distance between the user and the attacker.

Typically, only approximate distance, which is rounded to kilometers, is displayed to the user, but we confirmed that in some dating apps, exact distance in meters is contained in the packets sent by the server. The fourth application's server only sends well-known locations that are close to the queried user, and this makes it difficult for the attacker to acquire the exact location of the queried user. However, when victim's location does not change, the attacker can repeatedly change his GPS information before sending requests and repeatedly collect distances to the victim. If the attacker obtains three or more distances between arbitrary coordinates and the victim, the victim's exact location can be calculated by triangulation.

Table 1. Our analysis results with five dating apps in the anonymous user environment.

	A app	G app	T app	D app	N app
User profile	(✓) (random)	✓ (sequence)	✗ (random)	✓ (sequence)	✗ (sequence)
Location information	-	✓ (distance)	✓ (distance)	(✓) (special area)	✓ (distance)
User credential	-	-	-	-	-
Chat messages	-	-	-	-	-

✓ = applicable; ✗ = not applicable; (✓) = partially applicable

5.3 Co-located Attacker

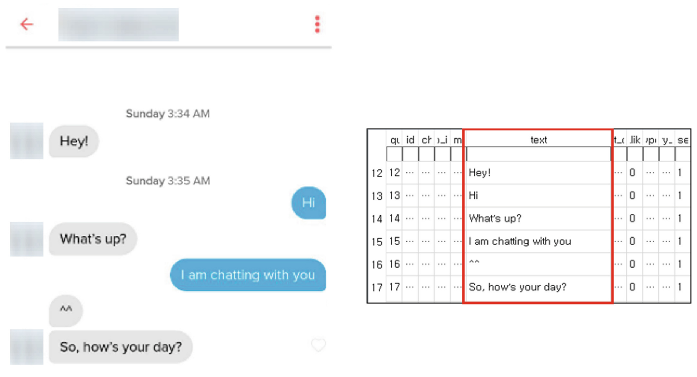
User Credential. Most dating apps maintain users' login status using an authentication token or a cookie stored in the shared preferences when performing post-authentication. To analyze the security of user credentials, we extracted and analyzed the storage of the dating app and analyzed the cookie content in the packet. As shown in Table 2, four of the five apps that we analyzed store the credentials in shared preferences which is stored as an xml file in the */data/-data/[package name]/shared_prefs/* path. Furthermore, the fifth app stores the credentials as cookie which is stored as a database in the */data/data/[package name]/app_webview/Cookies* file. The co-located attacker can use the app, masquerading as the victim, by cloning the credentials to the attacker's device.

Chat Messages. Dating apps provide many functionalities for finding potential partners, and one of them is the chat operation. All dating apps provide chat capabilities, but it can be problematic if the contents of chats are easily viewed by attackers. As shown in Table 2, the results of the chatting analysis in five apps show that one of the apps stores the chat history in the database without any encryption. This is depicted in Fig. 2. This means that this application's chat contents can be obtained from the storage on the device. Two of the five apps use an external API. The two remaining apps use their own web services to provide chat functionality, but the chat room index is random, making them difficult to be extracted.

6 Countermeasures

In this section, we suggest several defense strategies to mitigate the privacy issues described in Sect. 3.

User Profile. Many mobile applications provide user authentication functionality. The authentication services assign user indexes to the users in order to authorize them. To prevent a user index from being guessed by attackers, a user index with high entropy should be used. For example, using a hash function to generate a user index with high entropy makes it difficult to guess the user index.


Sunday 3:34 AM

Hey!

Sunday 3:35 AM

Hi

What's up?

I am chatting with you

^^

So, how's your day?

	qu	id	ch	_i	m	text	t_	lik	pi	y_	se
12	12	···	···	···	···	Hey!	···	0	···	···	1
13	13	···	···	···	···	Hi	···	0	···	···	1
14	14	···	···	···	···	What's up?	···	0	···	···	1
15	15	···	···	···	···	I am chatting with you	···	0	···	···	1
16	16	···	···	···	···	^^	···	0	···	···	1
17	17	···	···	···	···	So, how's your day?	···	0	···	···	1

(a) Chat messages in the app. (b) Chat messages in the database.

Fig. 2. Leakage of chat messages.

Table 2. Our analysis results with five dating apps in the co-located attacker environment.

	A app	G app	T app	D app	N app
User profile	(✓) (random)	✓ (sequence)	✗ (random)	✓ (sequence)	✗ (sequence)
Location information	-	✓ (distance)	✓ (distance)	(✓) (special area)	✓ (distance)
User credential	✓ (shared pref)	✓ (shared pref)	✓ (shared pref)	✓ (cookie)	✓ (shared pref)
Chat messages	✗ (random)	✗ (external API)	✓	✗ (external API)	✗ (random)

✓ = applicable; ✗ = not applicable; (✓) = partially applicable

Location Information. Many mobile apps require users to grant GPS permission before location information of the users can be collected. However, if such location collecting function is misused by a service provider, an attacker may be able to obtain a victim's location information. Examples include functions that are responsible for providing the user's latitude and longitude coordinates, or providing an accurate distance when searching for nearby people. In order to prevent exposure of the location information, it is necessary to insert intentionally vague location information or rough information so that the attacker cannot find the exact location of the victim.

User Credentials. Most mobile apps use user credentials for post-authentication. Examples of such user credentials are cookies, tokens, and sessions. Reuse attacks are possible by cloning user credentials. To prevent reuse attacks, expiration time of user credentials can be shortened. However, by doing so, it is obvious that usability might be negatively impacted.

Chat Messages. Many apps provide chat services for communication between users. Such a chat service may be provided as a web service, or it may exist in a database system. When it is provided as a web service, a chat index is calculated in a similar method that is used to generate a user index. A chat index having high entropy should be used to prevent any chat history from being exposed. On the other hand, if mobile applications store chat history in a local database, it should be stored after being securely encrypted to avoid exposure.

The issues mentioned in Sect. 3.2 are almost standard coding issues, where app developers did not employ a security-by-design approach and violated best practices. Developers should try to reduce these coding issues, but there are many more difficulties. Previous research to teach better secure coding and to improve app security exist [20]. That research provides an interactive IDE-based security review tool to improve the security and privacy aspect of code written by developers.

7 Related Work

Most social networking service (SNS) applications including social dating apps provide various functions like searching for other users or discovering locations of other users. Sometimes these basic functions may be maliciously used by an attacker. Kim et al. [16] showed that Facebook's search functionality could potentially be misused to leak users' sensitive personal information on a large scale. Since the search function of Facebook enabled users to search for other people using their phone numbers, a large amount of user profile data could be collected by searching for consecutive phone numbers. Carmen et al. [9] and Hoang et al. [15] discussed that mobile apps using location information of users cannot guarantee the user's location privacy because the user's location can be estimated by using trilateration. Especially, Hoang et al. [15] analyzed if three dating apps (Jack'd, Grindr and Hornet) were protecting their users' location privacy securely. They found that the attacker could figure out the location of other users in all the three apps using a trilateration method, even when the apps provided the location-hiding option as a countermeasure to location estimation. In this paper, we comprehensively analyzed not only these location providing and search functions but also other functions provided by dating apps such as acquaintances blocking and chatting.

Farden et al. [13] investigated privacy risks using forensic analysis, focusing on storage in mobile devices in particular. The goal of this research was to confirm what data was stored in mobile devices using forensic analysis. It was shown that sensitive data such as usernames, profile pictures, sent messages and authentication tokens could be recovered by an attacker in some dating apps. Patsakis et al. [21] performed an analysis to test if sensitive personal information and location information could be retrieved by capturing HTTPS packets in some dating apps. In our work, we also analyzed how attackers can acquire user credentials and chat messages from dating apps. We showed that if an attacker can access the storage in the path of dating apps, then the attacker can access sensitive data.

Wondracek et al. [26] introduced a de-anonymization attack that exploits group member information on SNS. They showed that it was possible to identify a particular user from a group of users or identify possible candidates. They also showed that about 42% of users in the social network Xing who use groups could not be uniquely anonymized. Dating apps provide anonymization to prevent anyone from disclosing other users' information such as names, emails, or cell phone numbers that can be used to identify users. However, in this paper, we showed that one of the dating apps automatically extracted an e-mail address from other user profiles that were anonymized, which, in turn, can be used to identify other users.

Privacy issues concerning sensitive user data on android mobile applications were often discussed in some previous work [17,28,31]. These studies were performed with permissions or predefined policies. In our paper, we analyzed the privacy issues that could arise in the dating apps to identify more hazardous privacy leaks than the ones discussed previously.

Android analysis technique can be generally divided into static analysis [8,14,27,30], dymamic analysis [12,23,29] and hybrid analysis [24,25]. Most of these analysis frameworks are used for malware analysis. When used for identifying privacy issues, only limited information such as phonebook, basic mobile phone information (IMEI, USIMID) and location information can be analyzed for outflow. With these frameworks, it is difficult to find a privacy issue specific to dating applications similar to those discussed in this paper.

8 Conclusion and Future Work

In this paper, we examined sensitive assets in dating apps and how this critical information is being stored in mobile devices. We also categorized the privacy issues that may arise in dating apps into four categories (e.g user profiles, location information, user credentials, and chat messages). We confirmed that at least one of the four privacy issues occur in each of the five apps. Furthermore, we discussed potentially serious attack scenarios that may compromise users' privacy and security.

Privacy issues that we listed in this paper are not exhaustive, and there may be other privacy issues that we may not have considered. For instance, the location tracking feature of dating apps might be used to predict the movement of the user. Furthermore, personal user data in dating apps can be combined with other user data collected from various sources (e.g., social network services) that may lead to compromise of user privacy.

Our analysis methods are also applicable to other applications like instant messengers that store information such as chat messages, photos, contacts, and user profiles. Therefore, we plan to develop a generic framework to automatically analyze such privacy breaches on Android applications.

Acknowledgement. This research was supported by the MSIT (Ministry of Science, ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2015-0-00403) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

1. Android app ranking. https://www.appannie.com. Accessed 25 Mar 2018
2. Fiddler. https://www.telerik.com/fiddler. Accessed 4 July 2018
3. Frida. https://www.frida.re/. Accessed 4 July 2018
4. Online dating apps growing. https://www.statista.com/outlook/372/100/online-dating/worldwide. Accessed 4 July 2018
5. Tinder Hits $3 Billion Valuation After Match Group Converts Options. https://www.forbes.com/sites/stevenbertoni/2017/08/31/tinder-hits-3-billion-valuation-after-match-group-converts-options/. Accessed 4 July 2018
6. Tinder personal data. https://www.theguardian.com/technology/2017/sep/26/tinder-personal-data-dating-app-messages-hacked-sold. Accessed 4 July 2018
7. tPacketCapture. https://play.google.com/store/apps/details?id=jp.co.taosoftware.android.packetcapture. Accessed 4 July 2018
8. Au, K.W.Y., Zhou, Y.F., Huang, Z., Lie, D.: Pscout: analyzing the android permission specification. In: Proceedings of the Conference on Computer and Communications Security (2012)
9. Carman, M., Choo, K.-K.R.: Tinder me softly – how safe are you *Really* on Tinder? In: Deng, R., Weng, J., Ren, K., Yegneswaran, V. (eds.) SecureComm 2016. LNICST, vol. 198, pp. 271–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59608-2_15
10. Cho, J., Kim, D., Kim, H.: User credential cloning attacks in android applications: exploiting automatic login on android apps and mitigating strategies. IEEE Consum. Electron. Mag. **7**(3), 48–55 (2018)
11. Cobb, C., Kohno, T.: How public is my private life?: privacy in online dating. In: Proceedings of the 26th International Conference on World Wide Web (2017)
12. Enck, W., et al.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst. **32**(2), 5 (2014)
13. Farnden, J., Martini, B., Choo, K.K.R.: Privacy risks in mobile dating apps. In: Proceedings of 21st Americas Conference on Information Systems (2015)
14. Fuchs, A.P., Chaudhuri, A., Foster, J.S.: Scandroid: Automated security certification of android. Technical report (2009)
15. Hoang, N.P., Asano, Y., Yoshikawa, M.: Your neighbors are my spies: Location and other privacy concerns in GLBT-focused location-based dating applications. In: Proceedings of 19th International Conference on Advanced Communication Technology (2017)
16. Kim, J., Kim, K., Cho, J., Kim, H., Schrittwieser, S.: Hello, Facebook! Here is the stalkers' paradise!: design and analysis of enumeration attack using phone numbers on facebook. In: Liu, J.K., Samarati, P. (eds.) ISPEC 2017. LNCS, vol. 10701, pp. 663–677. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72359-4_41
17. Li, L., et al.: Iccta: detecting inter-component privacy leaks in android apps. In: Proceedings of the 37th International Conference on Software Engineering (2015)
18. Li, M., et al.: All your location are belong to us: breaking mobile social networks for automated user location tracking. In: Proceedings of the 15th International Symposium on Mobile ad hoc Networking and Computing

19. Lutz, C., Ranzini, G.: Where dating meets data: investigating social and institutional privacy concerns on tinder. Sage Social Media + Society (2017)
20. Nguyen, D.C., Wermke, D., Acar, Y., Backes, M., Weir, C., Fahl, S.: A stitch in time: supporting android developers in writingsecure code. In: Proceedings of the Conference on Computer and Communications Security (2018)
21. Patsakis, C., Zigomitros, A., Solanas, A.: Analysis of privacy and security exposure in mobile dating applications. In: Boumerdassi, S., Bouzefrane, S., Renault, É. (eds.) MSPN 2015. LNCS, vol. 9395, pp. 151–162. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25744-0_13
22. Shetty, R., Grispos, G., Choo, K.K.R.: Are you dating danger? an interdisciplinary approach to evaluating the (In)security of android dating apps. IEEE Trans. Sustain. Comput. (2017)
23. Tam, K., Khan, S.J., Fattori, A., Cavallaro, L.: CopperDroid: automatic reconstruction of android malware behaviors. In: Proceedings of the Network and Distributed System Security Symposium (2015)
24. Wang, S., State, R., Ourdane, M., Engel, T.: Riskrank: security risk ranking for ip flow records. In: Proceedings of the 6th International Conference on Network and Service Management (2010)
25. Wei, X., Gomez, L., Neamtiu, I., Faloutsos, M.: ProfileDroid: multi-layer profiling of android applications. In: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (2012)
26. Wondracek, G., Holz, T., Kirda, E., Kruegel, C.: A practical attack to de-anonymize social network users. In: Proceedings of the 31st Symposium on Security and Privacy (2010)
27. Yang, Z., Yang, M.: LeakMiner: detect information leakage on android with static taint analysis. In: Proceedings of the 3rd World Congress on Software Engineering (2012)
28. Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., Wang, X.S.: Appintent: analyzing sensitive data transmission in android for privacy leakage detection. In: Proceedings of the 20th Conference on Computer & Communications Security (2013)
29. Zhao, M., Zhang, T., Ge, F., Yuan, Z.: RobotDroid: a lightweight malware detection framework on smartphones. Citeseer J. Netw. **7**(4), 715 (2012)
30. Zhao, Z., Osono, F.C.C.: "TrustDroid^{TM}": preventing the use of smartPhones for information leaking in corporate networks through the used of static analysis taint tracking. In: Proceedings of the 7th International Conference on Malicious and Unwanted Software (2012)
31. Zhu, H., Xiong, H., Ge, Y., Chen, E.: Mobile app recommendations with security and privacy awareness. In: Proceedings of the 20th SIGKDD International Conference on Knowledge discovery and data mining (2014)