Who leaks my privacy: Towards Automatic and Association Detection with GDPR Compliance

Qiwei Jia, Lu Zhou, Huaxin Li, Ruoxu Yang, Suguo Du, and Haojin Zhu

Shanghai Jiao Tong University, China
{jiaqiwei,zl19920928,lihuaxin003,yangruoxu,sgdu,zhu-hj}@sjtu.edu.cn

Abstract. The APPs running on smart devices have greatly enriched people's lives. However, they are collecting personally identifiable information (PII) secretly. The unrestricted collection, processing and unsafe transmission of PII will result in the disclosure of privacy, which cause losses to users. With the advent of laws and regulations about data privacy such as GDPR, the major APP vendors have become more and more cautious about collecting PII. However, the researches on detecting privacy leakage under GDPR framework still receive less attention. In this paper, we analyze the clauses of GDPR about privacy processing and propose a method for PII leakage detection based on Association Mining. This method assists us to find many hidden privacy leakages in traffic data. Moreover, we design and implement an automated system to detect whether the traffic data sent by the APPs reveals users' PII. We have tested 509 APPs of different categories in the Google Play Store. The result shows that 76.23% of the APPs would collect and transmit PII insecurely and 34.06% of them would send PII to third parties.

Keywords: GDPR · Privacy Leakage · Association Mining.

1 Introduction

The widespread use of smart devices has led to the development of various APPs. While providing users with services, these APPs also collect users' personally identifiable information (PII) [1, 4]. Researches show that the proportion of APPs leaking personal data has increased exponentially in recent years, and significant numbers of APPs leak personal data to third-parties [5, 14].

In order to regulate the data collection, transmission and processing behavior of operators, many countries have enacted different laws and regulations, such as EU General Data Protection Regulation (GDPR), American California Consumer Privacy Act (CCPA) of 2018, and People's Republic of China Network Security Law. Among these laws and regulations, GDPR is considered as the most important change in data privacy regulation in 20 years [12, 20].

The GDPR reshapes the way in which data is collected, transmitted and processed. Compared with previous laws, it expands the material and territorial scope of protection, and is known as the most advanced and strict law on data protection and privacy. With the implementation of the GDPR, it is necessary to recheck the privacy leakage problem in APPs.

Existing detection methods for APP privacy leakage include the following two aspects: analyzing APPs and detecting traffic data. Analyzing APPs by static or dynamic methods often results in high false positives, and it is difficult to form an automated system [1, 4–8]. Detecting traffic data are mainly based on string search and regular matching. Although precise PII strings can be detected, there are still many potential privacy leakages if the APP developers deliberately modify the keywords in the traffic data to evade detection [2, 3, 9–11].

In order to mine hidden information which cannot be shown by simple string search and regular matching, we propose a novel detection method, association mining based on the iterative key-value pair matching, to detect the APP privacy leakage under the GDPR environment. The main idea of this method is that personal data such as personal identifier, device identifier and location information on a single terminal will not be changed in an experimental environment. Therefore, the associations between keys and values can be used to mine more indirect associations among different keys and discover the hidden information.

The contributions of this paper are summarized as below:

– We analyze the GDPR clauses regarding the data collection, transmission, and processing, and find out the differences from other privacy laws.
– We propose a novel detection method, association mining based on the iterative key-value pair matching, to explore the potential privacy leakage issues hidden in traffic.
– We implement a privacy leakage detection system based on association mining and detect the top 509 Android APPs in Google Play Store. The result shows that 76.23% of the APPs are collecting and transmitting privacy data insecurely, and 34.06% will send PII to third parties at least once;

The rest of the paper is organized as follows. Section 2 lists related works about privacy leakage detection. Section 3 gives the interpretation of GDPR and puts forward the subjects of study. Section 4 introduces the methodology. Section 5 gives the design and implementation of the privacy leakage detection system. Section 6 evaluates the APPs in Google Play by our system. Section 7 concludes the paper.

2 Related Work

2.1 The analysis of APP Programs

As is mentioned above, the static analysis is a method of analyzing the program file without executing it. The most popular static analysis techniques include dataflow analysis, symbol execution and so on. Using static analysis methods, we can reach each accessible branch of the program and track the data flow. However, in actual works, many branches in a program are actually unreachable, which may cause false positives, and the dataflow-tracking may cause path explosion, which will increase the number of calculations exponentially [1, 4–6].

Dynamic analysis means to analyze an APP when it is running on a real or virtual device. During the process of analyzing, the analyst may use hook functions and debug tools to observe APP behaviors and data transmissions. Sometimes, the analysts should modify or customize the systems, and a large number of operations rely on manual works. Therefore, it is difficult to develop an automated detection system, and the efficiency is relatively low [7, 8].

2.2 The analysis of traffic data

The detection of traffic data also falls into two ways: detecting traffic data from Internet Service Providers (ISPs) and from a single terminal. The former is usually a joint work completed by the analysts and the ISPs with techniques including string search, regular expression matching, and machine learning. The analysts aim to find personal data in traffic or to classify traffic packages as privacy-sensitive ones or not. Huge amounts of data can be acquired from the ISPs, but it is unable to distinguish if a package is generated by the specific APP, and the numerous invalid packages may bring many uncertain interferences [11].

Detecting traffic data on a single terminal seems to be a better way. The existing researches include TaintDroid developed by W. Enck et al [8], which provides real-time privacy analysis using Android's virtualized execution environment. J. Ren et al. implemented a cross-platform privacy leakage detection system Recon, and it shows privacy leakage during APPs running in a visual way [3]. I. Reyes et al. tested 5,855 children's apps on Google Play and found that 73% of them transmit sensitive data over the Internet [2]. The above works have promoted the development of privacy leakage detection on a single terminal. However, after analyzing plenty of traffic packages, we found that many of them contain hidden privacy leakages, especially customized or ambiguous keywords that can not be detected by previous methods.

2.3 The analysis based on the GDPR

As the GDPR was released recently, there are not many researches on it. Some articles combined with the GDPR to detect privacy leakages mainly have two ideas: one is to define the scope of the privacy according to the GDPR, then apply machine learning, string search and other methods to detect privacy leakages during the execution of the APPs. For example, WB. Tesfay et al. subdivided Privacy Sensitive Information (PSI) into 13 types such as Sexual orientation, Relationship, Emotions and so on, and then use the semi-supervised machine learning method to classify the unstructured texts from Twitter, and analyze whether they contain PSI or not [17]; the other idea is to check the APPs' compliance with a certain clause of the GDPR. P. Ferrara et al. designed a method of taint analysis to merges all sources of sensitive data, then reconstruct the data flow, and finally generate a report of the GDPR analysis [18]. F. Kammller took advantage of the Attack Trees to verify GDPR compliance and illustrated on the example of a healthcare IoT [19].

The first of the above two ideas does not actually interpret the GDPR, nor detect the specific clauses, and the second is essentially a static detection which has the problem of path explosion. Our work will start with interpreting the specific GDPR clauses related to privacy leakages in mobile, and apply them to make an automated detection of the APPs.

3 GDPR and Privacy Leakage Detection

The General Data Protection Regulation (GDPR) [20], passed by the European Union in April 2016 and came into effect on May 2018, is a regulation with the aim of strengthening personal data and privacy protection for residents of the EU. With the implementation of GDPR, many APP vendors have adjusted their privacy policies and become more cautious in collecting and processing privacy.

3.1 Scope and definition

Chapter 1 Article 3 of GDPR stipulates the territorial scope, it applies to the personal data controllers and processors in the EU, wherever they work. APPs storing or processing personal data of the EU citizens, regardless of whether they have businesses in the EU, must meet with the GDPR [20].

Chapter 1 Article 4 of GDPR defines several keywords. ‘Personal data’ includes the name, identification number, address, an online or social identifier of a natural person. By processing ‘personal data’, the individual’s performance, economic status, health, personal preferences, interests, reliability, behavior, location or whereabouts, etc., collectively called ‘profiling’, can be evaluated. In mobile, ‘personal data’ usually includes the following parts:

- Personal identifier (PI): name, gender, date of birth, email address, etc.
- Device identifier (DI): network, hardware information, phone number, etc.
- Location information: longitude, latitude, base station information, etc.

3.2 Data collection and secure transmission

Chapter 2 Article 6 of GDPR states that the legality of processing, including collection, storage, modification, transmission, etc., must first satisfy the condition that ‘the data subject has given consent’. When installed or running, the APP may apply for many unnecessary permissions and collect irrelevant users’ personal data, which violates the GDPR.

In terms of data transmission, Chapter 4 Article 32 stipulates that measures such as anonymization and encryption should be adopted to ensure security when processing personal data. If the APP transmits personal data using plain text or a simple encryption algorithm, users’ information would be easily stolen.

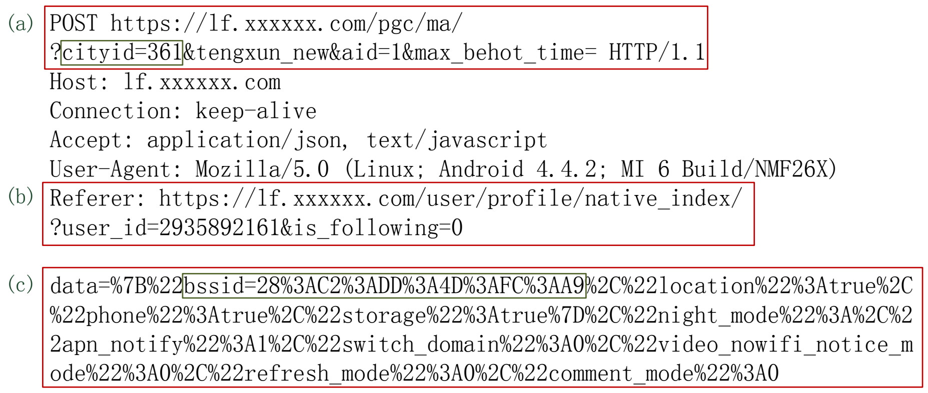
3.3 Third-party privacy disclosure

It is clarified in Article 13 of Chapter 3 that ‘when collecting personal data, the APP should provide with the identity details of the data controller, and the purposes as well as the legal basis for the processing, and the recipients of the personal data'. APPs may send data to third parties for reasons such as remote calls, information collection, etc., and this further increases the risk of privacy disclosure. From the terms of the GDPR, the behavior that APPs transmit personal data to third parties without users' explicit knowledge, violates the regulations of the GDPR.

4 Methodology

4.1 Data Extraction

In mobile, most of the APPs transmit data through the HTTP/HTTPS protocol. The substance of privacy leakage detection is analyzing specific HTTP/HTTPS packages in the network. Fig. 1 shows a typical HTTPS request package. Three sections in request packages, i.e., (a) the request line, (b) the Referer header field and Cookie header field, and (c) the request data, may contain a large amount of valid personal information, as they are used as the carriers of request parameters or APP data.



```
(a) POST https://lf.xxxxxx.com/pgc/ma/
    ?cityid=361&tengxun_new&aid=1&max_behot_time= HTTP/1.1
    Host: lf.xxxxxx.com
    Connection: keep-alive
    Accept: application/json, text/javascript
    User-Agent: Mozilla/5.0 (Linux; Android 4.4.2; MI 6 Build/NMF26X)
(b) Referer: https://lf.xxxxxx.com/user/profile/native_index/
    ?user_id=2935892161&is_following=0

(c) data=%7B%22bssid=28%3AC2%3ADD%3A4D%3AFC%3AA9%2C%22location%22%3Atrue%2C
    %22phone%22%3Atrue%2C%22storage%22%3Atrue%7D%2C%22night_mode%22%3A%2C%2
    2apn_notify%22%3A1%2C%22switch_domain%22%3A0%2C%22video_nowifi_notice_m
    ode%22%3A0%2C%22refresh_mode%22%3A0%2C%22comment_mode%22%3A0
```

Fig. 1. Example of HTTPS packages.

The contents of the Referer header field and the request header both consist of a domain name and several parameters. The format of parameters is key-value pairs like 'key1=value1&key2=value2&key3=value3&...', where the equal sign (=) is an assignment character, and the ampersand (&) is a connector. The key is a name defined by developers, while the value is its corresponding content. In most cases, the actual meaning of the key value can be inferred by the name.

The cookie field and request data contain the cache information and form information sent by the APP. Although having no fixed assignment characters and connectors, they usually appear as key-value pairs. Valid key-value pairs can also be extracted as long as the assignment characters and connectors are effectively recognized.

4.2 Association Mining

In general, a key can express its actual meaning, so analysts can examine whether the key carries personal data or not [10]. However, after analyzing a large amount of traffic data, we find that it is impossible to filter all the packages containing personal data through common keys. In some cases, the developer deliberately modifies the key in order to evade the detection. Thus, simple common keys matching cannot accurately infer whether a package carries personal data. For example, the key-value pair 'bssid=28%3AC2%3ADD%3A4D%3AFC%3AA9' in Fig. 1 is actually the MAC address rather than a Basic Server Set id. In this case, such APPs can leak a large amount of privacy without being noticed.

To address this problem, we propose the key-value association mining. Its main idea is that: (a) pre-set a collection of keys as a key set; (b) collect and analyze traffic data for a large number of APPs on a single terminal, extract all key-value pairs; (c) for each key in key set, extract the associated values in all key-value pairs to expand the value set; (d) for each value in the value set, extract the associated keys in all key-value pairs, expand the key set and remove duplication; (e) repeat (c) (d) until the key set and the value set no longer grow, and the matching is completed.

The main assumption of this method is that personal data such as the personal identifier, device identifier and location information on a single terminal in a session will not be changed. So the associations between keys and values are used to discover more indirect associations among different keys, and thus enlarge the coverage of privacy leakage detection. As shown in Fig. 2, the key 'latitude' finally matches a key-value pair of 'cityid=24' which can not be easily found through common methods.

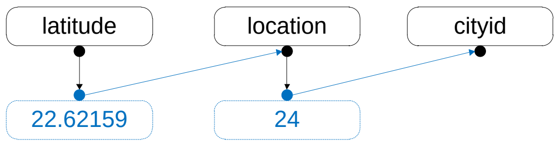


Fig. 2. The process of association.

5 System Design and Implementation

We design and implement our system prototype on Android OS, which is running on more than 85% smart devices. We setup our detection system on Android Emulators to automatically examine APPs' privacy leakage. The detection system includes three modules: APP Crawler, Automation Platform, and PII Detector.

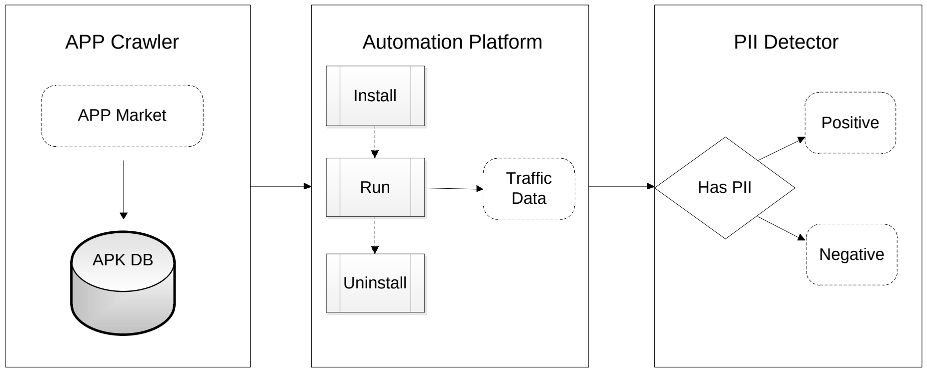


Fig. 3. Privacy leakage detection system.

5.1 APP Crawler

The APP Crawler is responsible for collecting APPs from APP stores (e.g., Google Play), and storing them by categories. Robustness and parallelism are considered when crawling, so techniques such as error handling and multi-threading are used in the implementation.

In our experiments, the Crawler starts with reading the APP category names and links on the main page of Google Play. Then for each category, it goes to sub-pages and downloads the top 20 APPs of each category. The automation was implemented by the Beautiful Soup library[1] in Python. In the end, we have downloaded 509 APPs of 27 categories from Google Play and saved them with their package names.

5.2 Automation Platform

The Automation Platform is responsible for installing, running and uninstalling the APPs. It would manipulate user interface for 15 minutes, capturing and saving the traffic data. The platform must ensure that only one APP is running at a time in order to eliminate interference from the other APPs. APPs connect to the Internet through a Man-in-the-Middle (MIM) proxy, while the traffic capture tool working on the proxy collects traffic data for further analysis.

APPs Runtime Emulation. We use Yeshen Emulator[2] in our prototype. Yeshen Emulator is a popular Android emulator that can customize a virtual Device id, SIM number, IMEI, IMSI and other private parameters of Android OS. It can also set virtual GPS data. ADB (Android Debug Bridge) debugging tools are integrated into Yeshen Emulator and we use the ADB Monkey tool to send some pseudo-random user events (such as clicking, returning, back to the desktop, etc.) to emulate UI operations.

[1] `https://www.crummy.com/software/BeautifulSoup/`
[2] `https://www.yeshen.com`

Traffic Data Capture. We capture traffic data by Fiddler2[3], which integrates the functions of MIM proxy, traffic interception, and HTTPS decryption. We only need to store the traffic data for the APPs being tested, so we want to discard the traffic data not generated from the emulator (e.g., traffic data from other hosts in LAN or physical machines) and traffic data generated by Android system APPs and pre-installed APPs in the emulator. So we set Fiddler2 to filter traffic packages in the network and only save packages by Yeshen Emulator. Then we run 'iptables firewall' through the ADB shell, set firewall rules to prevent the other APPs from connecting the Internet.

5.3 PII Detector

After previous steps, we now have a large number of traffic packages, which are composed of a request line, header fields, and request data. First we need to extract the valid data and convert them into key-value pairs. Then we generate the complete key set and value set, and detect whether a package leaks PII or not. Finally, we make statistical analyses for quantifying privacy leakages.

Pre-processing. We extract the three valid sections as mentioned above and divide them into key-value pairs by different connectors. The key-value pairs follow the pattern of 'key(assignment character)value(connector)key(assignment character)'. The request line and the Referer field use the equal sign (=) and the ampersand (&) as the assignment character and connector respectively, while the cookie field and request data have no fixed formats and characters. Some common characters we have seen are :=/ (assignment characters) and ,;&+— (connector). Therefore, we use a simple statistical method to solve the challenge, that is, counting the numbers of assignment characters or connectors respectively in the sections, and selecting the most ones.

Besides, there are many binary data and other invisible characters, as well as a lot of newline characters, brackets, quotation marks in a real traffic package. In addition, the generated key-value pairs contain many invalid data, such as random strings, domain names, etc. These data are filtered and removed in pre-processing as well.

Privacy Pairs Generation. We conduct association mining as described in Section 4.2 to generate more complete key set and value set and expand them until they cannot grow anymore. In order to avoid the explosion of associated keys/values that are added into the sets, we only expend frequent keys/values whose number of appearance is larger than a threshold (=3 in our experiment) in each round. This can avoid a large number of random keys/values that are not related to user privacy.

We set the initial key set with some common device identifier and location information keywords including 'phoneNumber', 'imei', 'imsi', 'androidid', 'mac',

[3] `https://www.telerik.com/fiddler`

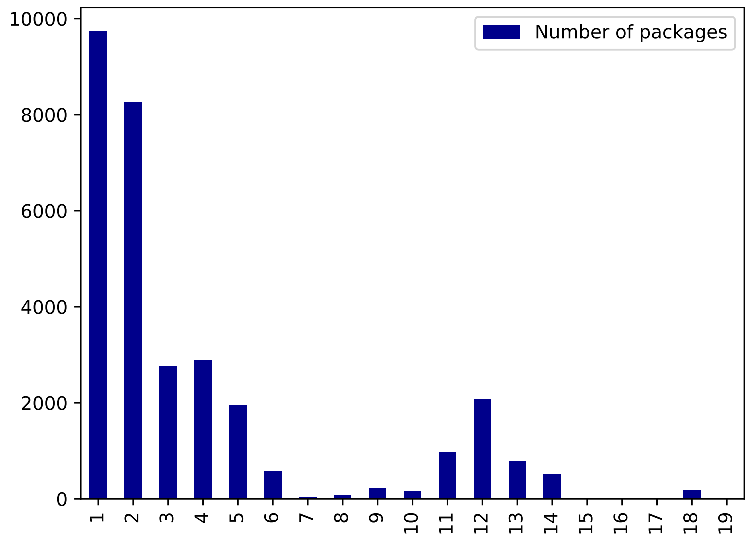


Fig. 4. The distribution of privacy-sensitive key-value pairs in packages.

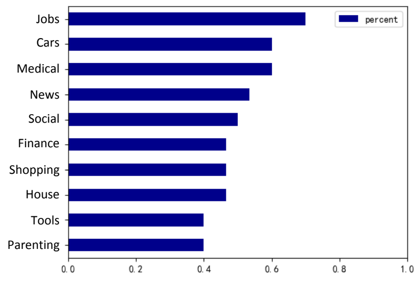
'longitude', 'latitude'. At the end of the expansion, many other keys related to PII or sensitive data in the key set, such as 'system_info', 'deviceid', 'devicename', 'providceId', 'city_i', 'areaId', 'mypos', 'adCode', are found with our methods in the experiment. Some keys/values that are not related to privacy may be included. In this case, we filter and remove values which are obviously not related to user privacy.

In the end, we analyze the privacy leakage of APPs by the key set and value set. When the key and the value of a key-value pair are both in the set, the pair is evaluated as a privacy pair. A package which includes one or more privacy pair is classified as a privacy-sensitive one.

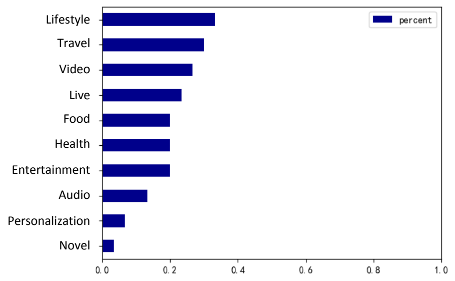
6 Evaluation

In our experiments, we have crawled 509 APPs of 27 categories from Google Play, captured 169,326 packages, and extracted 629,217 valid key-value pairs. The overall analysis result shows that the privacy-sensitive key-value pairs have appeared in about 34.0% of all captured packages and on average each one carries 3.716 pieces of privacy, as shown in Fig. 4.

Unconsented Data Collection. For the analysis of APPs, we find that 388 out of the 509 APPs transmit personal data over the Internet, accounting for 76.23%. These personal data are captured and transmitted in our automated detection system without 'the data subject's consent', so the APPs do not conform to Chapter 2 Article 6 of GDPR. This proportion is higher than 61.9% or 63.3% by the other existing methods [6, 16], which shows the effectiveness of our method. Among them, APP categories such as Jobs, Cars, Medical and News



(a) The top 10



(b) The last 10

Fig. 5. Categories ranking in transmitting personal data.

transmit more personal data, while Novel, Personalization, Audio and Entertainment APPs seem more secure, as shown in Fig. 5.

Unencrypted Personal Data Transmission. Device identifiers (e.g., android id, IMSI, etc.) are observed to be the most personal data collected by APPs. The APP vendors leverage them to identify devices and target ads to users. When transmitting personal data, more than 39% APPs use the unencrypted HTTP protocol, though decreased 9% in the past 2 years, which are considered insecure and violate Chapter 4 Article 32 of GDPR. The different personal data and their transmitting methods are shown in Table 1.

Table 1. The different personal data and their transmitting methods.

Type	**HTTP**	**HTTPS**	**Total**
Device ID	50408(23.56%)	65424(30.38%)	115832(54.14%)
Network Info	11534(5.39%)	30268(14.15%)	41802(19.54%)
Location	21622(10.11%)	34394(16.08%)	56016(26.18%)
Others	92(0.04%)	192(0.09%)	284(0.13%)
Total	83656(39.10%)	130278(60.90%)	213934(100%)

Third-party Data Disclosure. Sending personal data to a third-party without consent is likely to violate Article 13 of Chapter 3 of GDPR. We extract the SLD (second-level domain) from the Host header filed of a package, then judge whether the host is a third-party or not according to the attribution of SLD. We find that 173 of the 509 APPs have sent personal data to third-parties, accounting for 34.06%. The Keywords that are most concerned by third-parties are listed in Table 2.

Power of Association Mining. After analyzing the key set and value set generated in our association mining, we find that many different keys have the same values, which are their actual meanings, as shown in Table 3. Analysts would miss a lot of privacy leakages if they only search for sensitive key names in network traffic.

Table 2. The Keywords that are most concerned by third-parties.

Keyword	Frequency	Keyword	Frequency
imei	19204	device_type	7891
device_id	11955	imsi	6266
os	10644	lat	6095
os_version	9176	lng	6063
uuid	8921	mac	5610

Table 3. Actual meanings and related keys appeared in packages.

Actual meaning	Related keys
imsi	net_oper, sim_serial, iccid, AD9, deviceid
android_id	aid, deviceId, distinct_id, did, openudid, uuid
imei	uuid, deviceToken, deviceId, devkey, meid
mac	mac_address, device_id, bssid, m
longitude	lng, lng_pos, currentLng, x1
latitude	lat, lat_pos, mypos, y1, geoia, latlng, X1, Coordinate

7 Conclusion

While changing people's lives, the APPs running on smart devices are collecting our privacy. With the implementation of laws and regulations such as GDPR, this problem has not been greatly improved. Some APP developers are even deliberately concealing the transmission of users' PII in traffic data, and it is difficult to detect this case by previous methods. In this paper, we propose a privacy leakage detection method based on Association Mining, design and implement a detection system, then apply it to the APPs in Google Play Store, and the result is as we expected.

We should know that the key-value matching can extend the boundary of privacy leakage detection and reduce false negatives, but it also increases false positives to a certain extent. By filtering key set and value set we can slow down the occurrence of false positives, but it cannot be completely avoided.

In future research, Combining the static or dynamic detection method with the analysis of data flow to infer the actual meanings of keys, the effect may be further improved.

References

1. Rui, H., Jin, Z., Wang, B.: Investigation of taint analysis for Smartphone-implicit taint detection and privacy leakage detection. EURASIP J. Wireless Comm. and Networking 2016: 227 (2016)
2. Reyes, I., Wijesekera, P., Reardon, J., On, A.E.B., Razaghpanah, A., Vallina-Rodriguez, N., and Egelman, S.: "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. PoPETs 2018(3): 63-83 (2018)
3. Ren, J., Rao, A., Lindorfer, M., Legout, A., Choffnes, D.R.: ReCon: Revealing and Controlling PII Leaks in Mobile Network Traffic. MobiSys 2016: 361-374

4. Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N.M., Bellovin, S.M., Reidenberg, J.R.: Automated Analysis of Privacy Requirements for Mobile Apps. NDSS 2017
5. Nan, Y., Yang, Z., Wang, X., Zhang, Y., Zhu, D., Yang M.: Finding Clues for Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps. NDSS 2018
6. Li, L., Bartel, A., Bissyand, T.F., Klein, J., Traon, Y.L., Arzt, S., Rasthofer, S., Bodden, E., Octeau, D., McDaniel, P.D.: IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. ICSE (1) 2015: 280-291
7. Razaghpanah, A., Vallina-Rodriguez, N., Sundaresan, S., Kreibich, C., Gill, P., Allman, M., Paxson, V.: Haystack: In Situ Mobile Traffic Analysis in User Space. CoRRabs/1510.01419 (2015)
8. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P. D., Sheth, A.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones.ACM Trans. Comput. Syst. 32(2): 5:1-5:29 (2014)
9. Continella, A., Fratantonio, Y., Lindorfer, M., Puccetti, A., Zand, A., Kruegel, C., Vigna, G.: Obfuscation-Resilient Privacy Leak Detection for Mobile Apps Through Differential Analysis. NDSS 2017
10. Liu, Y., Song, H.H., Bermudez, I., Mislove, A., Baldi, M., Tongaonkar, A.: Identifying Personal Information in Internet Traffic. COSN 2015: 59-70
11. Xia, N.,Song, H.H., Liao, Y., Iliofotou, M., Nucci, A., Zhang, Z., Kuzmanovic, A.: Mosaic: quantifying privacy leakage in mobile networks. SIGCOMM 2013: 279-290
12. Greengard, S.: Weighing the impact of GDPR. Commun. ACM 61(11): 16-18 (2018)
13. Book, T., Bronk, C.: I see you, you see me: Mobile advertisements and privacy. First Monday 21(3) (2016)
14. Li, H., Xu, Z., Zhu, H., Ma, D., Li, S., Xing, K.: Demographics inference through Wi-Fi network traffic analysis. INFOCOM 2016: 1-9
15. Mann, C., Starostin, A.: A framework for static detection of privacy leaks in android applications. SAC 2012: 1457-1462
16. Zhang, D., Guo, Y., Guo, D., Wang, R., Yu, G.: Contextual approach for identifying malicious Inter-Component privacy leaks in Android apps. ISCC 2017: 228-235
17. Tesfay, W.B., Hatamian, M., Serna, J., and Rannenberg, K.: PrivacyBot: Detecting Privacy Sensitive Information in Unstructured Texts. ICICS 2018: 156
18. Ferrara, P., Olivieri, L., Spoto, F.: Tailoring Taint Analysis to GDPR. APF 2018: 63-76
19. Kammller F.: Attack Trees in Isabelle. ICICS 2018: 611-628
20. General Data Protection Regulation(GDPR), `https://gdpr-info.eu`.
21. GDPR FAQs, `https://eugdpr.org/the-regulation/gdpr-faqs/`