A survey on key technologies of privacy leakage detection for Android platform

Fu Yu¹, Lv Xuming, Gao Xiao, Liu Lin, Luan Jingzhao
State Grid Liaoning Electric Power Co., Ltd., Shenyang, Liaoning 110006, China
¹0086-13609827088, lnepfy@163.com

ABSTRACT. With the rapid advancement of the mobile internet, smartphones have been the essential tools for communication and entertainment. In recent years, Android, developing quickly, has been the largest mobile operating system in the world. However, due to mass private data generated by Android intelligent devices, privacy disclosure is a serious problem in security for the Android platform. Based on the relevant background, this thesis indicates the problems of privacy protection that the Android platform faces, summarizes the detecting techniques for privacy disclosure on the Android platform at the present stage, and introduces the research status and progress with the aim of offering a reference for relevant practitioners and researchers.

1. INTRODUCTION
With the rapid development of the mobile internet, smartphone has been a kind of indispensable tool for communication and entertainment in public life. Especially when the functions of smartphones are enhanced, smartphones, to some extent, replace other function devices such as camera, VCR, and music player, which means that smartphones can not only meet the needs of communicating with others by telephone, but also be applied in our daily life like social contact, recreation, investing in stocks, shopping online and mobile payment. For this reason, there are mass private data of users in smart mobile devices. If the personal information is disclosed, users’ daily life will be seriously infringed[1]. For instance, disclosing the geographical location information may give rise to the infringement of users’ personal safety and disclosing the payment information like bank account may result in a loss in money.

Android system was released by Google on November, 2007. Since it was released, developing dramatically, Android system has been one of the hottest operating systems of smartphones in the world, attracting a large number of developers to develop App programs on its platform. However, unlike Apple’s App market in a fence pattern, Google play, the App market created by Google, lacks strict detecting procedures to detect those App programs entering the market. Meanwhile, a lot of the third-party App markets increase the potential security risk of mobile phones and worsen the problem of users’ privacy disclosure[2]. Due to several kinds of unsafe factors such as Android system and the App market, the private data in the Android devices is easy to be stolen and utilized by malicious attackers. Generally speaking, there are four kinds of ways disclosing privacy on the Android platform: (1) Private data disclosure between Android Apps; (2) Private data disclosure in Android Apps; (3) Privacy disclosure during the process of data transmission (network transmission)’ (4) Data collected by sensors causes privacy disclosure[3].

In order to protect private information of Android users, more and more researchers has focused on the study on detection methods and technologies of privacy disclosure on the Android platform. According to the relevant background of the Android platform, this thesis points out the outstanding problems about privacy protection that need to be solved. Additionally, the thesis also makes selective
analysis on the major detecting methods of Android privacy disclosure in the present stage, including
the static analysis and the dynamic analysis.

2. The Outstanding problem of privacy protection on the Android platform
On the aspect of privacy protection of users, the Android platform takes a series of security measures to
protect the privacy security of users. In order to protect smartphones from losing or being stolen, android
provides the lock screen and full disk encryption for data partition of the Android system by using master
keys which are randomly generated. Therefore, if users lose their smartphones, the privacy information
storing in the smartphones cannot be disclosed unless the person involved knows the password of the
lock screen. As for the Apps operating on the Android platform, Android system also provides a series
of security measures such as the sandbox mechanism, the permission mechanism, and the signature
mechanism. In terms of the permission mechanism, Android system, based on the permission
management, provides a set of privacy protection mechanism to protect private data for users. In the
privacy protection mechanism, App developers must make a clear statement of the required permissions
in the document AndroidManifest.xml and strictly follow the security specification set by Android
system to avoid these permissions being disclosed to the malware. Before installing the new App, users
should carefully read all permission statements of the App, and only by being accepted by users, the
App can be installed in the devices of users. Android system should make sure that only with the relevant
permissions, Apps can be allowed to have access to private data or related functions. However, common
users do not know whether the permissions applied by Apps are safe or not. As for Apps, Android system
also provides the protection mechanism of digital signature. When users install an App, the system can
detect the digital signature of the App. The system will detect whether the App was once installed in the
smartphone or not, which means whether the name of the program package about to install is the same
as that of the installed program package or not. If the names are matched, the system will treat the App
about to install as the upgrade version of the installed App and then replace it. If the names are
unmatched, the system will stop installing. Although some protection mechanisms such as sandbox
mechanism and SLINUX mechanism can protect users’ privacy to some extent and restrain the
rampancy of the malware, these measures show the weakness when they tackle endless black
technologies and attack techniques. Under this circumstances, in the study of how to improve the privacy
protection of Android system, there are some outstanding problems to solve.

(1) Privacy data hiding and its deniability. Because the Android system only protects the inner
privacy data by password of lock screen, and most people adopt easy and simple patterns to lock the
screen, when users’ smartphones are lost and users are hijacked, it is easy for others to get the passwords
which may result in the inner privacy disclosure. If the system can encrypt to hide the private data in the
smartphones, it can fill the gap of the weak protection of passwords of lock screen. By this way, even
the password is cracked, or users are forced to give the password, the hidden private data can be well
protected and users can deny the private data.

(2) The safe and reliable system operation environment. Currently, most of Apps on the Android
platform contain service components, running in the background. This will cause a large number of
unreliable Apps (such as entertainment or game software) operating when users use sensitive Apps like
Alipay. Much worse, there are some malware gaining root privilege by making use of system bugs.
Under this condition, sensitive Apps will be attacked by the malware so that the account information
will be disclosed. Additionally, the private data storing in the smartphone may be stolen by the malware.

(3) Permission control of fine grit of internal storage. At present, with the price decline of the flash
memory, the capacity of internal memory of smartphones rapidly increases to meet the needs of user
information storage, which makes device manufacturers gradually no longer to support the removable
TF memory. As for users, their personal data can only be stored in the internal memory. App developers
usually store some key data in the memory (emulation memory). However, in order to have access to
the internal memory and all files in the emulation memory, Apps just need to apply for the permissions
of SDCARD_R and SDCARD_W. This is the main way of privacy disclosure, also the main target
attacked by the ransomware with mobile encryption.
3. The Static Detecting Technique for Privacy Disclosure

As the common analysis technique in the field of mobile security, the static analysis technique for detection in Android means that instead of a code of an App under detection being operated, some techniques like lexical analysis and syntax analysis are used to analyze and detect the source code, decompiled code or intermediate code of the App to make sure whether the App will give rise to the privacy disclosure or not. The static detecting technique stresses on static state, so the main way during the process of detection is analyzing and scanning the decompiled code. There are two situations in the static detecting technique of Android Apps. The first one is to analyze and detect the Dalvik bytecode files or Smali files made by decompilation. Another is to analyze and detect the Java code made by decompilation. The key to distinguish these two situations is the different objects of analysis. The static analysis technique can be divided into two analysis methods, one of which is based on the feature code while the other is based on the syntax and semantic of the code.

(1) The analysis method based on the feature code is to generate the feature code for malicious act by analyzing the malicious App which is known to all. In fact, analyzing the Apps which were not analyzed before means to match the feature codes. Compared with the dynamic analysis technique, it is unnecessary for the static analysis technique to use the extra resources such as sandbox and virtual machine, and before the operation of malware, it can detect and scan. For this reason, it is characterized by high efficiency and low risk. However, the result of this kind of analysis method relies on the completeness of the feature database. With the new variation viruses appearing constantly, the feature database will be larger, which will not only lead to the detecting efficiency decline, but also increase the expense on maintaining the enlarged feature database.

(2) The analysis method of code semantic is the detecting method based on the syntax and semantic of the code, which scans the code of malware and acquires its implementation as well as the semantic relation according to the features of Java syntax and semantic to avoid some interference like code obfuscation. The basic process of this kind of detecting method is to analyze the framework of APP codes and malware codes, and then extract the syntax and semantic features of both codes. After that, the semantic association diagrams will be set up based on these features and the similarity of two diagrams will be calculated.

To sum up, the general process of the static analysis for Android Apps is shown as Figure 1. It can be seen that the static analysis method is to scan and analyze the control flow and data flow to ensure whether it will lead to the privacy disclosure or not when the Android Apps are not under operation. AndroidLeaks[2] developed by Clint Gibler et al., ScanDrok[4] developed by Fuchs et al. and LeakMiner[5] developed by Yang Min et al. from Fudan University all use the static analysis method. However, these analyzing and testing systems mainly aim at the scanning and analysis of Java source code. At present, most of Android Apps are not open source. Therefore, firstly, there are many measures like decompiled operation for Apps and reverse engineering should be taken to extract the features of the Apps and get the information like relation of function call and binary sequence. In literature [6], FlowDroid provides more accurate information for the transformation of life cycle of Apps by carefully analyzing the whole life cycle of the Apps. Through developing DroidBench on the Android platform, FlowDroid can test the accuracy of taint detecting methods. The result shows that its accuracy is higher than AppScan and Fortify.

![Decompiled APK files](image)

Figure 1 General process of static analysis

4. The Dynamic Detecting Technique

Some malicious codes may hinder the decompiled process by some methods such as encryption and decryption, code obfuscation and packer, which requires the dynamic detecting technique to avoid the problems. The dynamic detecting technique can detect the App based on its operation in the sandbox or the virtual machine. The technique can be divided into sandbox security mechanism and dynamic taint
tracer technique. The sandbox mechanism serves as a mechanism running in isolation. When it detects the privacy disclosure of Apps, it needs to detect the privacy leakage source of the system and API of the privacy leakage point in real time. Once the App invokes these interfaces, records should be taken in real time. Because the sandbox mechanism needs additional resources and the monitoring points are discontinuous, there lacks logical relationship and the context. In the analysis of privacy disclosure made by the dynamic taint tracer technique, users’ sensitive information are often marked as the tainted data and then the spread track will be traced. If the tainted data are found to be leaked by message, the internet or bluetooth, the App is judged to exist the privacy leakage problem. The detailed process is shown as Figure 2.

![Figure 2 Stain analysis process](image)

TaintDroid [7], the privacy disclosure testing tool developed by Enck et al. uses the dynamic taint tracer technique early on the Android platform. According to TaintDroid, it firstly marks some privacy leakage sources (such as short messages, geographical location information and IMEI), mainly adding the 32-bit tainted label near the location of each data variable. Secondly, it traces the taints of coarsness (the level of IPC and file messages) and fine grit (the level of Dalvik virtual machine instructions). The last step is to detect API at the exit of the internet, files and short messages, including detecting the private messages about to send to judge whether they are the tainted data, recording tainted private data and the label type, and removing the tainted label before sending messages. However, the disadvantage of TaintDroid is that the process of the analysis detection requires human intervention rather than automatically detect privacy disclosure.

At present, most of safe-defend software adopt the dynamic detecting technique [7]. For example, in order to detect privacy disclosure, Paranoid Android [8], developed by Georios et.al, copies the operation condition of Android Apps, checks and kills the virus on the remote server, and detects the taint spread by taking advantages of the functions of the virtual machine including recording and playback. Homyack et.al upgrades TaintDroid as the tool which can filter the private information when the information is disclosed. It can largely protect the users’ private data without affecting the operation. The literature [10][11] belowdesigns and finishes the sandbox system on the Android platform based on the detection of basic dynamic act. Additionally, it can also monitor the operating act of Apps in the frameworks of different levels. The DroidScope system in literature [12] makes a dynamic detection for malware on the Android platform by recreating SalvikVM of the Android system.

5. Conclusions

Combined with the relevant background of the Android platform, the thesis points out the outstanding problems about privacy protection which needs to be solved by the Android platform, and summarized the research status based on the private leakage detecting techniques on the Android platform, including the static detecting technique and the dynamic detecting technique. In General, various privacy leakage detecting techniques have been the focus of researchers, and they needs to improve to fill the gaps even though they grow rapidly. With the fast execution and the comprehensive coverage of codes, the method of static analysis yet cannot solve the problems like code obfuscation and high rate of false alarm while
the method of dynamic analysis can better solve the problems of the static analysis, but it also exists some problems like insufficient coverage rate of codes and low operation. In conclusion, how to combine the advantages of the static analysis with the dynamic analysis to make a mixed analysis is the trend of privacy protection on the Android platform in the future.

ACKNOWLEDGMENTS
Sponsored by State Grid Corporation of China Technology Program (Contract No. 2019YF-20)

REFERENCES
[1] Y. T. Jang, S. E. Chang, and Y. J. Tsai, “Smartphone security: understanding smartphone users' trust in information security management,” Security and Communication Networks, vol. 7, no. 9, pp. 1313-1321, 2014.
[2] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale." pp. 291-307.
[3] T. Heer, O. Garcia-Morcho, R. Hammen, S. L. Keoh, S. S. Kumar, and K. Wehrle, “Security Challenges in the IP-based Internet of Things,” Wireless Personal Communications, vol. 61, no. 3, pp. 527-542, 2011.
[4] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, Scandroid: Automated security certification of android, 2009.
[5] Z. Yang, and M. Yang, "Leakminer: Detect information leakage on android with static taint analysis." pp. 101-104.
[6] C. Fritz, S. Arzt, S. Rasthofer, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, “Highly precise taint analysis for android applications,” 2013.
[7] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones,” ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, pp. 5, 2014.
[8] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: versatile protection for smartphones." pp. 347-356.
[9] S P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications." pp. 639-652.
[10] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection." pp. 281-294.
[11] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for android malware detection." pp. 1011-1015.
[12] L. K. Yan, and H. Yin, "DroidScope: Seamlessly Reconstructing the {OS} and Dalvik Semantic Views for Dynamic Android Malware Analysis." pp. 569-584.