An Android Application Vulnerability Mining Method Based On Static and Dynamic Analysis

Wang Chao[1], Li Qun[1],Wang XiaoHu[1],Ren TianYu[1],Dong JiaHan[1],Guo GuangXin[1],Shi EnJie[1]
1.State Grid Beijing Electric Power Research Institute, Beijing, China
29681987@qq.com, 664668506@qq.com, w007wxh@163.com, 280284492@qq.com, 249333924@qq.com, ggxqxh@sina.com, shi_enjie@sina.com

***Abstract*—Due to the advantages and limitations of the two kinds of vulnerability mining methods of static and dynamic analysis of android applications, the paper proposes a method of Android application vulnerability mining based on dynamic and static combination. Firstly, the static analysis method is used to obtain the basic vulnerability analysis results of the application, and then the input test case of dynamic analysis is constructed on this basis. The fuzzy input test is carried out in the real machine environment, and the application security vulnerability is verified with the taint analysis technology, and finally the application vulnerability report is obtained. Experimental results show that compared with static analysis results, the method can significantly improve the accuracy of vulnerability mining.**

Keyword—Android APP; Vulnerability mining; Static analysis; Dynamic analysis

I. INTRODUCTION

In recent years, the development of applications based on Android platform has shown a trend of rapid growth, and the accompanying security issues have attracted more and more attention. Due to the lack of security awareness of developers, there are various loopholes in the program design[1-3]. At the same time, the domestic Android application market is chaotic and lack of effective unified management mechanism, the application security quality is difficult to be guaranteed, and the number of security vulnerabilities is increasing year by year.

Therefore, Android application vulnerability mining has attracted more and more attention from researchers. Generally, vulnerability mining of Android applications can be divided into static analysis and dynamic analysis. The paper [4] analyzed a large number of applications through static analysis, and found that there is a data flow analysis to improve vulnerabilities between applications, which can easily lead to privilege elevation attacks.. The paper [5] performed reverse analysis on the application, statically analyzed the data flow of the application code, tracked the taint of sensitive data, obtained the propagation path of sensitive data in the application, and played a guiding role in the detection of application vulnerabilities. TaintDroid [6]first proposed the use of dynamic taint analysis technology in the Android system to detect privacy leaks. TaintDroid is a system-level dynamic taint analysis technology, so there is no need to modify the application, and private data transmission within and between applications can be recorded and tracked by TaintDroid. With the development of security hardening technology, static analysis technology increases the difficulty of vulnerability mining because it cannot decompile application source code[7-10]. And the application does not have an actual operating environment, so there may be false positives and false negatives in the vulnerability results. In order to solve the problem of static analysis, the paper improves the mining accuracy by introducing fuzzy dynamic testing technology based on static analysis.

II. Vulnerability mining process

Based on the analysis of the advantages and limitations of static analysis and dynamic analysis of android applications, a vulnerability mining method based on the combination of static and dynamic analysis is proposed. This method first disassembles the code and analyzes the code, and uses type inference and type constraints to obtain test cases based on the vulnerability security rules. Then, the test cases are mutated by using the security rule base, and fuzzy test is carried out by

using the method of taint tracking. The flowchart is shown in Fig. 1.

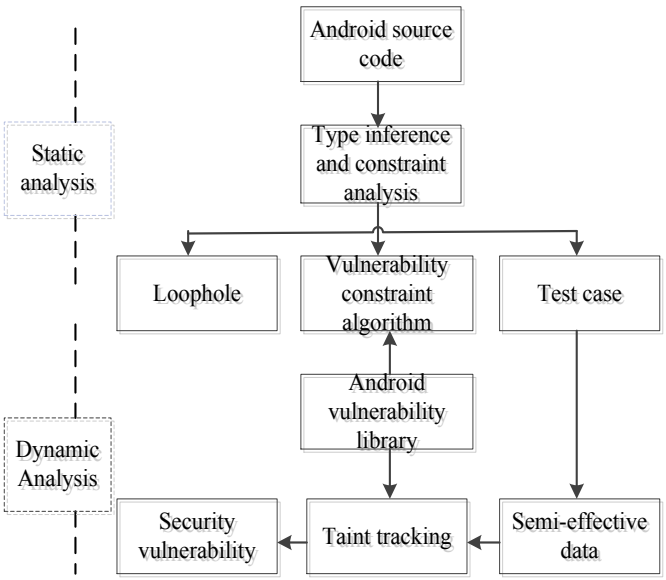

Fig. 1 Vulnerability mining process

III. Static analysis

A. Static analysis process

Android vulnerability static analysis reads and analyzes the source code of the program through automated tools to detect key points of possible security vulnerabilities in the program. The static analysis process is shown in Fig. 2.

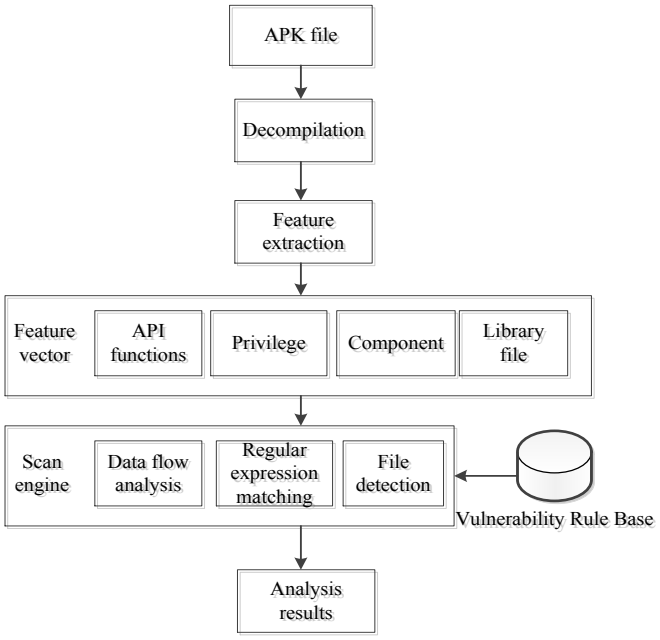

Fig. 2 Static analysis process

First, the Android application is decompiled and preprocessed, and the application source file is obtained. Then, the application feature vector is obtained by feature extraction. Finally, according to the vulnerability rule base, the static analysis results are obtained by using the scan engine for analysis and matching.

B. Decompilation

The purpose of decompilation is to obtain the source code of the application, such as Java source code and AndroidManifest.XML. First, the APK file is decompressed to obtain AndroidManifest.AXML, DEX file, resource file, etc. Then, the dex2jar tool is used to decompile, convert the Dex file into a Jar file, and use the jd-gui tool to decompile the obtained file to obtain the application Java source code. The AndroidManifest.AXML file can be reversed to get the AndroidManifest.XML file by using the AXMLPrinter2 tool. Decompilation process is shown in Fig. 3.

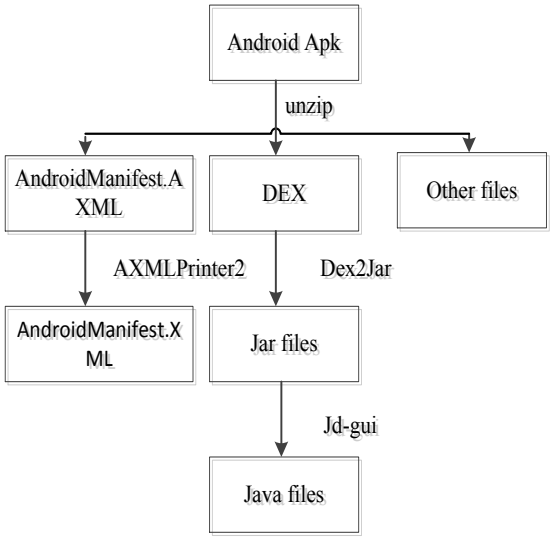

Fig. 3 Decompilation process

C. Vulnerability rule extraction

1) Component exposure

If the android application component is exposed, attackers can hijack its component for related operations by constructing a message object that conforms to the definition of the application component. The generation condition of component exposure is that when applying for component in the manifest file, the exported property is set to "true", or when the exported property is not set to "false", the

application sets the intent filter for the component, which may cause the component to be called by other applications. The detection of such vulnerabilities is mainly to match the feature sets such as component and permissions obtained by static analysis.

2) SQL injection

It is commonly used in Web programs. Related research has found that Android applications can also introduce SQL injection vulnerabilities when executing unsafe SQL statements. Android applications commonly use SQLLite to store data. The most common SQL vulnerability is that applications use raw SQL statements to operate. When SQL statements have external input parameters, SQL injection attacks may be launched.

3) Webview remote code execution

The addJavascriptInterface method in the Webview component is used to implement the interaction between local Java and JavaScript, but this function does not restrict the method call, which causes an attacker to call any JAVA class, and eventually causes JavaScript code to perform arbitrary attacks on the device.

4）Certificate weak check

In the customized x509trustmanager subclass, the server-side certificate is not verified, and any server-side certificate is accepted by default, which may lead to security risks and may cause malicious programs to bypass certificate verification by using man-in-the-middle attacks.

5) Allow Backup vulnerabilities

When the allow Backup property value in the AndroidManifest.xml file is set to true (the default is true), the user can back up the application data through adb backup and export the data stored in the application, causing the leakage of user data. So you need to set the android: allow Backup flag to false.

6）Shared Preferences storage risk

The Shared Preferences file does not use the correct creation mode, which may be accessed by other applications and read and write operations may be performed, which may cause the leakage of sensitive information. It is usually necessary to avoid using the MODE_WORLD_READABLE and MODE_WORLD_WRITEABLE modes to create Shared Preferences files.

IV. Dynamic analysis

A. Dynamic analysis process

The idea of dynamic analysis is to verify the results of vulnerability analysis based on static analysis to improve the accuracy of vulnerability mining. First, test cases are generated based on the static analysis results, and then test data is executed on the terminal where the application is installed, and taint analysis is performed to monitor the abnormal behavior of the application to obtain dynamic analysis results. The dynamic analysis process is shown in Fig. 4.

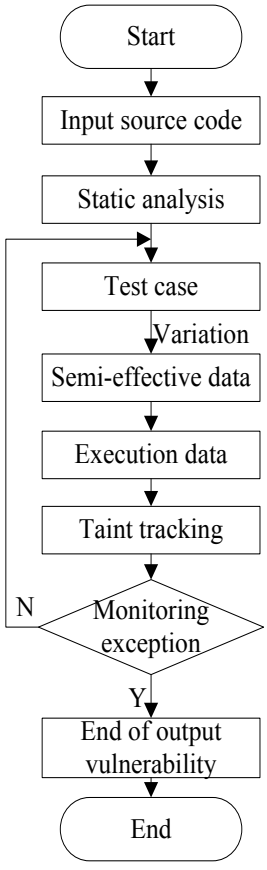

Fig. 4 dynamic analysis flowchart

B. Taint analysis

Taint analysis is a technique for tracking and analyzing the flow of taint information in a program. In vulnerability mining, the data of user interest is first marked as tainted data. The tainted attributes of these tainted data are inherited and propagated during the execution of the program. Then, by tracking the information flow related to the tainted data, it analyzes whether these data will affect some key program operations, and then analyzes and judges whether there are program vulnerabilities.

1） Taint initialization

Taint initialization is mainly to mark the taint. The taint mark contains two behaviors: first, find out where the taint information is generated in the program, and second, mark the taint information. Taint identification can be achieved by monitoring specific system functions. Such as fgets (), fread (), recv (), etc. The external data obtained by these functions should be set as tainted data. Regarding tainted marks, the point where the taint information is generated is called the source point. Correspondingly, the rules for identifying source points are called source point rules.

2） Taint spread

The use of specific rules to track and analyze the spread of taint information in the program is called taint tracking. At the binary level, it refers to the use of certain algorithms to resolve different instructions on the basis of tainted marks to solve the problem of transmission of tainted marks.

3） Taint check

Taint check refers to the establishment of specific rules to determine whether the transmitted pollution data violates the rules. If so, it is considered that there may be vulnerability. Generally speaking, these rules are set on the key operations performed by the program. For example, once polluted data is detected as the destination address for jumps (jmp family instructions), calls (call, ret), and data movement, or other operations that cause the EIP register to be filled with polluted data, it will be considered an illegal operation.

4） Taint removal

If the data that is not polluted is assigned to the memory area or register marked as the dirty state, the dirty state of the target memory or register will be cleared. By using binary interpolation technology, before each machine instruction is executed, the actual content of each operand of the instruction is obtained, and whether it is polluted by the input data is queried. Then the corresponding operation of spreading and removing the taint is determined according to the specific type of instruction.

V. Experimental analysis

In order to verify the effectiveness of the vulnerability mining method, 290 applications were downloaded from third-party application markets for experimental verification. The application scan engine is developed in Python, and dynamic analysis is performed through a USB connection for real machine testing. The test platform is Huawei P20. Common types of security vulnerabilities were selected for verification in the experiments. The experimental results were compared with static analysis methods, and the final detection accuracy was judged by manual analysis. The experimental results are shown in Table I.

Table I. Experimental results

Vulnerability type	**Detection rate（%）**	
	Static analysis method	**The method in the paper**
Component exposure	100	100
SQL injection	78	95
Webview remote code execution	86	90
Certificate weak check	98	95
Allow Backup vulnerabilities	80	100
Shared Preferences storage risk	87	100
Total	88	97

It can be seen from the experimental results that compared with the static analysis method, this method can improve the vulnerability detection rate for security vulnerabilities such as SQL injection, Webview remote code execution, allow backup vulnerabilities, and Shared Preferences storage risk. On the

one hand, due to the case of shell reinforcement in Android application, the automatic static analysis tool failed to reverse decompile, which led to the failure of further feature matching, resulting in a high rate of false positives. On the other hand, the static analysis method could not simulate the actual application running environment, resulting in a high rate of false positives in some vulnerabilities detection.

VI. CONCLUSIONS

. The paper designs and implements a dynamic and static vulnerability mining engine for the android platform. From the perspective of combining static analysis and dynamic analysis, the security issues of Android platform applications are mined. The experiment proves that the stability and availability of the system can be maintained under a large number of samples, the scanning task can be completed in a short time, and the false positives rate can be kept at a low level. The next step is to optimize the application of reverse analysis technology and expand the scope of application of analysis framework.

REFERENCES

[1]. Qing si-han. Research Progress on Android Security[J]. Journal of Software, 2016, 27(1):45-71.

[2]. Zhang Yuqing, Fang Zhejun, Wang Kai,et al. Survey of Android Vulnerability Detection[J]. Journal of Computer Research and Development.2015,52(10):2167-2177.

[3]. Zhang Yuqing,Wang Kai, Yang Huan，et al．Survey of Android OS security[J]．Journal of Computer Research and Development, 2014．5l(7)：1385-1396(in Chinese.

[4]. Tang junwei, Liu Jiazhen, Li Ruixuan,et al. Android application vulnerabilities static mining technology[J]. J. Huazhong Univ. of Sci. & Tch.(Natural Science Edition),2016(44):20-24.

[5]. Yue Hongzhou, Zhang Yuqing, Wang Wenjie, et al. Android Static Taint Analysis of Dynamic Loading and Reflection Mechanism[J]. Journal of Cpmputer Research and Decelopment, 2017,45(2):313-327.

[6]. WANG Jian-wei, JIN Wei-xin, WU Zuo-shun, Taint Droid-based Dynamic Taint Detection Technology[J]. Communications Technology,2016,49(2):221-226.

[7]. DONG Guowei,WANG Meilin, SHAO Shuai ,et al. Android application security vulunerability analysis framework based on feature matching[J]. J tsinghua Univ (Sci & Technol),2016,56(5):461-467.

[8]. YANG Le， YANG Wen-jun .Research on vulnerability detection data based on Android[J]. 2019,35(4):25-28.

[9]. Li Fenglong, The Design and Realization of Android Application Vulnerability Excavation Engine Based On Dynamic and Static Scanning Technology[D].JiNan,ShanDong University.2016.

[10]. Reaves B, Bowers J, Iii SAG,et al, *droid:Assessment and Evaluation of Android Application Analysis Tools[J]. ACM Computing Surveys,2016.49(3):55.