A Novel Android Application Penetration Analysis Method

Hao Zengshuai, Meng Leizi, Zhan Xiong, Wang Jie, Yu Jianbo

Global Energy Interconnection Research Institute, Beijing,
102209, China

Keywords: Android application; defect analysis; penetration test

Abstract

As the Android craze spread through the world, even the rapid development of mobile technology can not prevent applications from vulnerability abuse. The existing detection system mostly applied static analysis techniques to single application, which make a limited standard of accuracy and efficiency. In this work, we propose a novel penetration model. First of all, we categorize the Android applications in repository. Android applications in each group are detected with static analysis and dynamic analysis. We construct a sample set under test after summarizing defect collection. Second, the sample set is taken penetration test by malicious application. Then we extract the effective attacking API sequence from malicious applications. In the end, we obtain the threat trigger model for this category of applications using machine learning.

1 Introduction

Android as a new generation of intelligent device based on Linux system is widely used in many fields, such as smart homes, smart phones etc. Ever since the first generation of Android smartphone since has been launched in 2008, Android smartphone develops rapidly because of its powerful function and excellent performance. Android smartphone in people's life, work and learning plays a more and more important role. However, due to the openness of Android system, there are more security problem appears during it brings convenience. According to iiMedia team about mobile Internet investigation report, more than 90% of the android applications suffer from different degrees of attacks, especially the financial application and game application, a lot of them have been infected by a large number of virus.

Android application defects analysis has become a very important research topic, and it gains large attention from many security research institutions and the scholars. The existing defects detection is mainly divided into two kinds: static analysis and dynamic analysis.

Dynamic detection need run the software or plug-in. And the behaviors in the process of running determine the presence of security defects [3]. Dynamic detection technology mainly includes the taint tracing method, fuzzy test [2], and the stack trace method. The sandbox mechanism based on Android system [10] studies the dynamic behavior of the application in real-time monitoring system and the output log. Enck and other scholars [1] achieve a dynamic detection system-TaintDroid, based on Android stain tracking system. A dynamic detection system based on Linux kernel is proposed in the literature [4], which detects the kernel layer and the API call while the runtime. The literature [9] proposes a fuzzy test method for Android application. The dynamic detection has a higher accuracy, but the detection time is longer, and the cost is higher.

Different from the dynamic detection, the static detection doesn't need to run the program. Static analysis is mainly aimed at the source code of the application or the intermediate code which is obtained through the decompilation. Through the mature detection rule or the defect mode, we can locate the application defect automatically. The literature proposed a data flow analysis of the authority detection method to detect the privacy disclosure of multiple applications aimed to defend the attack on Android application permissions. Literature [7] using the method of static detection, aiming at the establishment of the Android application, it tries to build a CFG and search the potential danger path. There are many limitations such as high False alarm rate and too much intermediate data, so the static detection is suitable to be used as an auxiliary method.

Aiming at the shortage of Android application software security defect detection technology, the lack of the trigger conditions and study of protective methods of defects, this paper proposes an Android Application Penetration Defect Analysis Method. The purpose of this paper is to use the new defect detection technology to improve the detection efficiency, to find more security defects, to avoid the malicious attacks.

The structure of this paper is as follows: the first chapter describes the overall framework of the system. The second chapter describes all key technologies which the system contains and involves. The third chapter describes the Validation results of the system, and the fourth chapter summarizes the work of this paper.

2 Architecture Design of Detect System

The architecture design of this system is shown in the following fig.1.

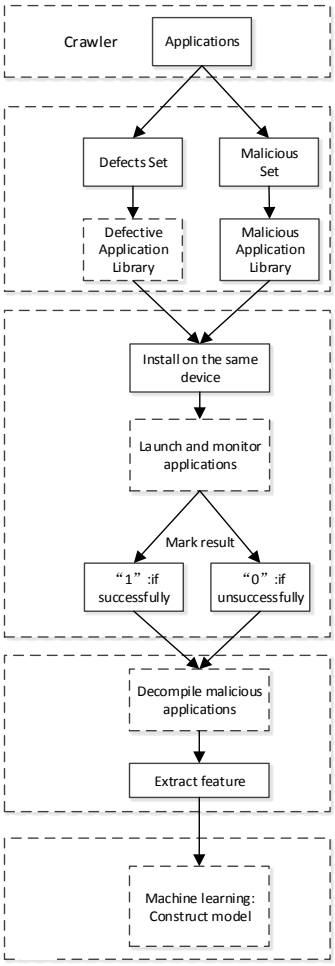

fig.1: Android application defect penetration system architecture diagram

The whole system is divided into five parts: application acquisition and classification, construction of defective application repositories, defective application penetration test, attack feature extraction of malicious applications and threat model construction.

2.1 Application Acquisition and Classification

According to the general classification method in network application store, we classify the Android application by their functions. Refer to the classification method in Google Play, 360 Application Market, MM market and so on, we classify the application into two classifications, one is software, the other is game. And each group has many small groups as shown in the following Table I.

Category	Specific Name	
Software	System tool	Video and audio
	Digital music	Online shopping
	Social applications	Browser
	Financial management	Applications for children
	Communication	…
Game	Shooting game	Strategy game
	Chess game	Online game
	Leisure game	Action game
	Role game	Sport game
	Racing game	…

Table I: Android Application classification table.

Then by means of web crawler, we crawl applications from the online store. According to the category, application is saved in the database. The crawler gets the application on the target network application store through the web crawler. Web crawler can automatically download the contents of the HTML page information. It has the characteristics of high automation and good maneuverability. It can meet the requirements for application in this paper. The process of crawler module is shown in the following fig.2.

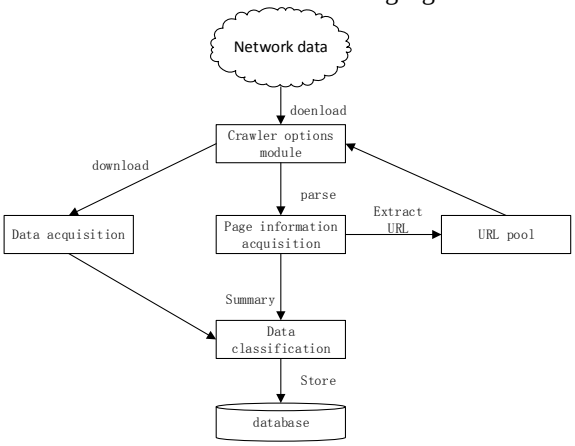

fig.2: process of acquisition and classification of applications.

In the databases, the application under analysis is saved according to category.

2.2 Construction of Defective Application Repositories

According to the category in section 2.1, we perform defect analysis on applications. And we find existing loopholes and defects in the application by static detection. Then for each category we sum up the defects, according to the defect sample collection we construct the database of specific defects, and it will work in later penetration analysis.

First, we define the application category in the databases $\mathrm{P} = \{A_1, A_2, \ldots A_\mathrm{n}\}$, A represents a category of all applications, P represents defective application repository, n represents the number of category in defective application database.
Second, Aimed at each category A, we summarize this category secure collection $S_i = \{B_1, B_2, \ldots, B_n\}$, S_i represents safety defect sets for class i applications, B_j represent the jth defect, m represent the number of total security defects in the current application category, according to the database and the definition secure defects collection, we construct the appropriate defect collection.
According to the application of a certain category, defect collection is defined as Si, the number of defect type is defined as m. we arrange the defects in the set and consider the influence of the relationship, then acquire the defect subset Qi. Therein, the number of the defect subset which has just one kind of defect is C_m^1, in other words, the number of it is m. And the number of the defect subset which has two kinds of defect is C_m^2. And so on, the number of the defect subset which has m kinds of defect is 1.

There is $1+ C_m^1 + C_m^2 + \ldots + C_m^m$ kind of defect in the constructed defective sample database, they are under penetration test.

2.3 Defective Application Penetration Test

First we collect malicious programs, and build a malicious program database. By using the malicious application, we make penetration test for defective application. While testing, we note down the real time detection of mobile phone and application status, then find out whether malicious program can effectively attack the defective applications, and count the result.
Application penetration test flow chart is shown in the following fig.3:

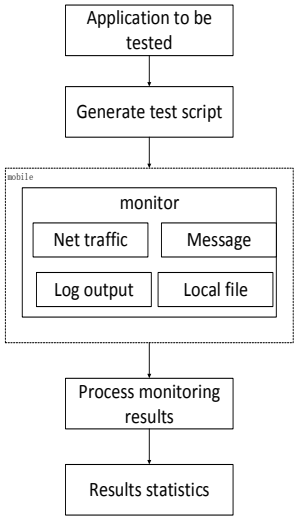

fig.3: Penetration test

Application under test includes the application of defect samples and malicious applications. Defect sample applications gather from the defect sample database established in this paper, and malicious applications gather from the database which laboratory has accumulated for many years, the number of malicious applications meet the needs of this paper. Only if we get enough applications and analysis, we can make the results more reliable. Then, we classify the malicious application in accordance with the application from a certain defect, and save them on each category.
Test script generation: According to the detection task of the platform, applications under test will be measured to generate the corresponding automatic test script in order to realize the whole process of automated testing and improve the detection speed[8]. The main function of this part includes the initialization of the detection environment, the establishment of communication with mobile terminal, the installation and unloading for applications, and the post-back from the execution, etc.
Runtime monitoring. Real time monitoring mainly include: SMS data, network traffic of send and receive, local data storage, file access behaviour, access to the distribution of the process, and the local resource usage, etc. These conditions can reflect the operation of the application in the mobile terminal. Therein, the mobile terminal is the main scene of the whole test process. It's convenient to build a real-time monitoring of the running state in application on the terminal.
Monitoring result analysis. We analyse the obtained result, and mark a simulated attack state (network connection, text messages sending, file or data deleting) to describe the security threat of the system. And then the state is stored as a set. If the defect samples of malicious applications can successfully attack 80%, it is judged that the malicious application can pose a threat to the type of application. The following work will extract the features of these malicious programs, and then use them as training samples for machine learning.

2.4 Attack Feature Extraction of Malicious Applications

For a certain category of applications, malicious application can attack in various ways. Many scholars, at home or abroad, have brought up many attacking methods on Android application. However, these methods are not systematic. In this paper, we have sorted and classified the attacking methods which helps to assess the extent of the defect. In this section, we first studied the attacking methods of malicious applications, and then extract features on basis of attacking methods.
We extracted features from source code. Actually the features are API sequences. The detailed process is shown in fig.4. First, we decompile the application to get the source code; we filter out the key API information from source code including caller/called API and system API. Then, we compare the key API to the list that already exist. If the match is successful, the API sequence will generate feature sample set.

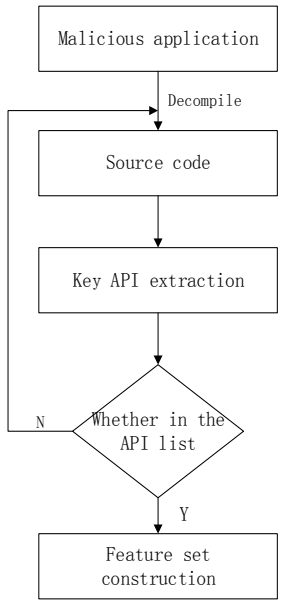

fig.4: Attack feature extraction

2.5 Threat Model Construction

In this paper, we obtain the corresponding malicious application set with flag "1" for each application category. This malicious application set has a great threat to this type of applications. In order to predict the the threat level of unknown application for each category, we construct the threat model on the basis of malicious application set. The model can accurately determine whether an unknown application is a threat to each class of application.

According to the attack feature extracted in section 2.4, we construct and train feature vectors. And we use SVM to perform machine learning. The entire process is shown in fig.5.

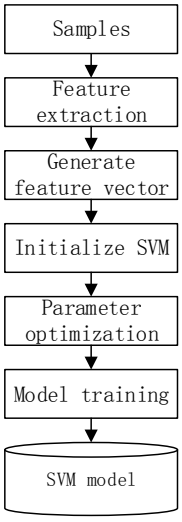

fig.5: SVM model construction

（1） First, we extract features from malicious application including API sequence;

（2） Then, we use the API sequences to construct the feature vectors and initialize the SVM;

（3） We will select different kernel functions to train the SVM and validate the output by cross validation method, repeat until the results are best;

（4） Finally, we will store the trained model in the database.

Next, we will look at the process of machine learning in more detail. We will extract API sequence $\{A_1,\cdots,A_m\}$ of malicious application for each class of defective application. We construct a data item $\{A_1,\cdots,A_m,\mathrm{r}\}$ combining the result of penetration test. All the malicious application data items constitute the feature set S.

The above feature need be mapped to a specific space. Therefor, we define $|S|$ dimensional vector[6] and eigenvector function $\delta(\mathrm{x})$. We obtain Equation (1).

$$\delta: x \to \{0,1\}^{|S|}, \delta(x) \to \left(I(x,s)\right)_{s\subset S}. \qquad (1)$$

Feature of application x is mapped to $\delta(\mathrm{x})$ and each feature correspond to 0 or 1. If application x contains feature s, $I(x,s)$ will be 1. Otherwise, it will be 0.

According to the equation above, the mapping location of sample in vector space has the characteristics as follows: The points with similar features in the vector space are close to each other, and the features with different features are far away from each other [5]. Therefor, we determine whether an application is threatening according to its position in vector space.

In this paper, the machine learning of malicious applications is carried out according to SVM above and we can obtain the threat model for each category of application. Threat model can effectively analyze the defect of application and reveal which API sequences can post threat to existing applications on mobile.

3 Core Technologies Research

3.1 Android Application Defects Analysis

3.1.1 Pre-processing Module

Its major function is to get configuration information and source code of application by decompiling. This module performs lexical analysis to source code, and generate abstract syntax tree, providing the basis for the follow-up module.

We get the AndroidManifest.xml and classes.dex by extracting the APK file. And we access to application-related information, including package name, version, component information by parsing xml file. We generate abstract syntax tree by parsing the smali files which are decompiled from dex file. Control Flow Analysis Module

Control flow analysis will extract information of ASTs and then form control dependence to support the following data flow analysis. control dependence graph was formed by sequence and jump relation of sentences in smali file. Meanwhile, jump branch was mainly directed by instructions in smali, so the key to obtain the control dependence graph is analyzing instructions in smali file.
The method used to parse source code of application is based on the theory of graphics. First, we transfer the code into a control flow graph. Then, we transfer the control flow graph into post dominator in tree. And we will get the control dependence graph according to the dependencies between post dominators.

3.1.2 Data Flow Analysis Module

Data flow analysis uses the data flow equation to iterate the data on the basis of the control flow graph. Data flow analysis Data flow analysis collects the semantic information of the program from the source code. The semantic information at different points of the program is linked by a set of simple equations. We can know behaviors of an application without running it on a device or a simulator. The purpose of data flow analysis is to calculate all the possible values of variables while the application is running. We need traverse all instructions in the control flow graph, and compute the set of constant values of each instruction iteratively.

3.1.3 Defects Analysis Module

This module can be divided into four parts, database of defect rule, resolving and configuration of rules, detection engine and results output.
(1) Defect rules are descriptions of a certain defects. The rules are stored in the database based on a unique file format.
(2) Resolving and configuration provide rules read from database for detection engine and control the parameters of output.
(3) Detection engine is the core of this module. It analyzes the control flow and data flow according to the rules provided in the previous section.
(4) The output is responsible for storing the results into database which is generate by detection engine.

3.2 Android Application Penetration Test and Defect Triggering

This module is responsible for monitoring the dynamic behaviour of application and assessing the threat of malicious application for defective application. The whole process can be divided into three stages.
(1) Preparation stage: According to the category of the application, we extract the application information from defective application library and malicious application library and set corresponding test tasks.
(2) Trigger stage: According to the test tasks, we set the test script, initialize the mobile device. The program will install defective and malicious application to mobile device following scripts and monitor behaviours of applications. Then the program will output the results and uninstall applications preparing for next test.
(3) Analysis stage: Based on the results of trigger stage, determine whether a malicious application can attack a defective application successfully. And we will mark malicious applications, which attack successfully, for each defective application.

4 Experiments and Results

4.1 Applications Sample Data

In order to ensure the diversity of samples, we download 13592 Android applications from several web app stores using the crawler server. And we store applications into the database according to their categories. The distribution of applications in three web app stores as shown in Table II.

App store	Count of application categories	Count of applications
YingYongBao	24	6147
Wan Doujia	29	4230
Mobile Market	40	3215

Table II: Application distribution data
We analyze applications and count up the numbers of defects for each category. The result of social applications is shown in fig.6. The result shows that denial of service has the highest share of the defects, followed by key management and SQL injection.

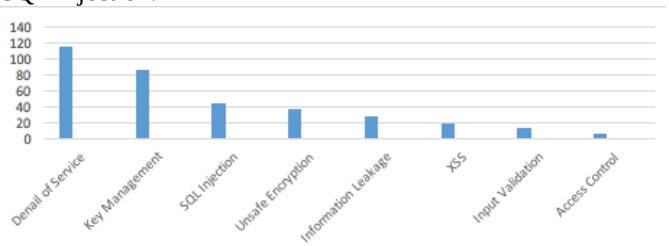

fig.6: Defects statistics of social application

4.2 Penetration Test Results

According to the category of applications, we got different categories of applications from database, and attacked those samples by applications in malicious application library. We recorded the results of the malicious apps attack, whether malicious apps attack successfully. A part of category applications test results are shown in fig.7 below.

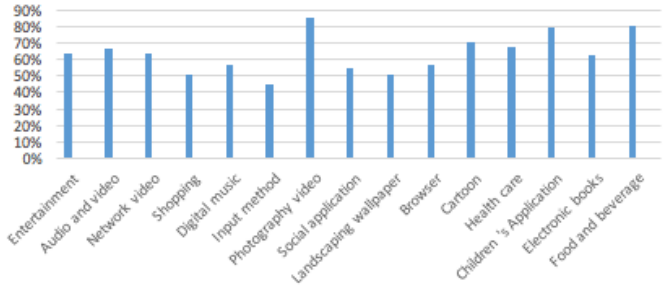

fig.7: The percentage of successful attacks on some categories of apps

4.3 Machine Learning Test Results

In this paper, we train sample and get trigger model by using SVM in sklearn Python module. We achieve the corresponding function by calling interfaces of the module. The support vector machine will use the SVC algorithm in SVM module. And The Poly kernel function was used to perform the non-linear mapping of the sample features. How to select proper kernel function?

In this paper, we respectively used RBF, Linear, Poly kernel functions and compared the accuracy of classification. Through experimental verification stage, we used ss_validation.cross_val_score function to perform cross validation. Detailed experimental results as shown in Table III.

It turned out that Linear kernel function cost longest training time and had least accuracy; Poly kernel function are most correct and were more stable than RBF. Moreover, Poly has global characteristics, which can ensure the edge point to the influence of kernel function.

	Linear	Poly	RBF
Accuracy /%	81.7	86.3	85.4
Training Time/s	1248	961	710
Test Time/s	410	332	275

Table III: Comparison of different kernel functions

Next, we are going to calculate the parameters of Poly function by using distance search method. We got the best result by cross validate samples, and the best penalty factor C is 350, kernel parameter x is 0.4175.

According to the detect results, we classified the features of malicious applications and acquired threat model of each category by the machine learning.

4 Conclusion

The traditional Android application defect analysis targets only a single application, detects security flaws in android applications using rules or models. As we all know, defects in the application are ubiquitous. The traditional defects analysis has the very limited protection to the mobile intelligent devices because of the lack of threat triggering condition, even it can detect the security flaws. Therefor, this paper presents a security flaw analysis method for mobile applications, then designs and implements a system of Android application penetration test, which provides a comprehensive means of detection and gives out threat triggering condition.

This paper has a shortcoming. Many Android applications are protected by code obfuscation and reinforcement. We can not get source code of those applications during pretreatment. And this brings big challenges to defects analysis. How to deal with those applications, which will become a key work from now on.

Acknowledgements

The acknowledgement for funding organisations etc. should be placed in a separate section at the end of the text.

Thank you for your cooperation in complying with these instructions.

References

[1] Chen, Yanling. "ADVANCEMENT OF THE STUDY ON FUZZY TESTING." *Computer Applications & Software*,**28(7)**, pp291-295, (2011).

[2] Enck, William, Machigar Ongtang, and Patrick McDaniel. "On lightweight mobile phone application certification." *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, (2009).

[3] Enck, William, et al. "A Study of Android Application Security." *USENIX security symposium*. pp.21-36, (2011).

[4] Fu, Jianming, et al. "A static detection of security defects between inter-components′communication." *Journal of Huazhong University of Science & Technology*, (2013).

[5] Narayanan, Arun, L. Chen, and C. K. Chan."AdDetect: Automated detection of Android ad libraries using semantic analysis." *IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing* ,**18**, pp.1-6 21-24, (2014).

[6] Westyarian, Y. Rosmansyah, and B. Dabarsyah. "Malware detection on Android smartphones using API class and machine learning." *International Conference on Electrical Engineering and Informatics*, **21**, pp.294-297, (2015).

[7] Yang, Bo, et al. "Method of Android Applications Permission Detection Based on Static Dataflow Analysis." *Computer Science* ,**39(z3)**, pp.16-20, (2012).

[8] Zeng, Li Kun, Q. B. Tang, and D. Niu. "Analysis the Security of Components in Android Application." *Software*, **35(3)**, pp147-151, (2014).

[9] Zhang, Mei Chao, F. P. Zeng, and Y. Huang. "Fuzzing Testing Based on Vulnerability Database." *Journal of Chinese Computer Systems*,**4(4)**, pp.651-655, (2011).

[10] Zhang, Y., et al. "Survey of Android OS security." *Journal of Computer Research & Development*, **51(7)**, pp.1385-1396,(2014).