1

The paper of our group is PATDroid, and the team members are 康金梦 詹睿 马驰 张金琛

2

Today's presentation will introduce the following parts to you in order according to the structure of the paper: (读目录)

3

The first is the introduction

4

At present, with the dynamic permissions system is introduced in the Android, the system allows the user after the installation of an application can also be granted to and revoked this program need to use the mobile phone privileges, testers are difficult to test the application correctly.

One solution that comes to mind is manual testing. Obviously, this method is very troublesome.

You can also use a machine to highlight all possible combinations of permissions, in which case.

Another option is to grant and revoke permissions at random. this approach does not thoroughly test the behavior of the application.

Therefore, the paper developed PATDroid to solve this problem, which is an automated method to effectively test Android applications while considering the impact of permissions on their behavior. Our results show that the PATDroid significantly reduces test effort and enables comparable code coverage and fault detection capabilities to thoroughly test applications with all permissions combined.

5.

The paper makes the following contributions:

• **Theory**: To the best of our knowledge, the first approach that considers the dependencies between a program, its test suite,and access control model for the reduction of testing effort;

• **Tool**: A fully automated environment that realizes the approach for Android programs, and made available publicly [10];

• **Experiments**: Empirical evaluation of the approach on a large number of real-world android apps demonstrating its efficacy.

6-7

For example，We use a simplified version of an Android app, called Suntimes,to motivate the research and illustrate our approach. For instance,wefoundthatTest#2 requires only Location permission,as the code executed by this test does not require access to capabilities guarded by other permissions. As a result, this test can only be executed twice—with and without the Location permission—rather than the 16 times required under the exhaustive scenario.

8-9

Throughout the paper, we call the apk containing the test suit and testing libraries as Test Harness App (THA), and the apk of the main app as App Under Test (AUT).

The figure depicts an overview of PATDroid, consisting of four major components. PATDroid first identifies those parts of AUT that could be exercised by the test cases embedded in THA .

THA analyzer determine the widget used in each test case (shown in the diagram for TW), the AUT analyzer determine permissions necessary for the execution of each block in the AUT (if any) and operation of these blocks of widget accessibility (that is EWP). Finally, Interaction Detector integrates the output of static and dynamic components and generates relevant permissions for each test case.

11

Android applications contain several methods called entry points are identified by the Dynamic App Analyzer component. PATDroid first automatic test given AUT, and in every possible at the beginning of the entry point of the application into the recorder, the recorder can be realized by the Android framework of a particular interface (such as onOptionsItemSelected, onResume, etc.). PATDroid then runs the entire test suite on the guidance application with any permissions set. Finally, the logs obtained through the detection of the application entry point are processed to capture the execution entry point for each test case. Generated by the output of this phase, called TE, is a set of tuples ⟨ test, entryPoint ⟩, one of the first element is to test identifier, the second element is in the process of test execution entry point.

12

In the case of low granularity, the dependency is obtained by static analysis in the entry point method. To statically track the dependencies between AUT and THA, pat-droid parses application input, or GUI widgets, which are the target of actions performed by test scripts.

In order to find the used widgets and make our approach test-framework-agnostic, the paper adopted the Android compatible data flow analysis framework FlowDroid. When solving the problem of data flow, THA analyzer generator output TW, it is a set of tuples, where the first element is to test identifier, the second element is a widget, it is a test performed by the action of target.

13

Run with any permission Settings, the dynamic application analyzer section explores the AUT code that each test can execute

Then PATDroid leverages AUT Analyzer to statically examine all parts of the code that could be exercised by each test.

The AUT analyzer receives TE as input and produces EWP as output. In general, the output is a set of tuples, each containing three elements.AUT Analyzer’s main procedure is summarized in Algorithm 1.(给老师看一眼，自己看看，组织语言，以备老师提问)

如果问的话：

The analysis procedure performs several steps to generate the output.Initially,PermissionAnalysis sub-procedure(line2)identifies the required permissions for executing each statement, if any, for all of the app’s entry-point methods exercised by the test suite. 随后，在第3行中调用WidgetAnalysis过程来确定由每个Widget控制的语句。

The WidgetAnalysis procedure is then called in line 3 to determine the statement that is controlled by each Widget.

14

Defined in each method, for a given AUT permissions analysis process by performing procedures in point analysis (summarized in algorithm 2) capture all permissions necessary for the execution of the method, called the permissions (PS). In the first step, the permission analysis builds a call graph (CG) for the entire application (line 2). Like this graph, it's suntimes, and in this graph, it's called implicitly by a dotted line. For example, the method getfixhelper.getfix () starts an AsyncTask, or GetFixTask, by calling the execute () interface. Therefore, the task class's doInBackground () method is called indirectly by the Android framework.

**（点击出现算法，自己看一下，组织语言，以备提问）**

15

左边的算法

For this purpose, a trimmed version of inter-procedural control-flow graph(ICFG)is constructed first(in line3)

Afterwards, the gen set is populated through iterating over every statement of each method (lines 4–11). Finally, the algorithm traverses the ICFG T in a breadth-first search manner and propagates the widget information through the graph.

右边输出的文字：

Combining the outputs of Permission Analysis and Widget Analysis sub-procedures, the main procedure generates the final output of AUT Analyzer component EWP.

16

Interaction Detector procedure, summarized in Algorithm 4, iterates over the three input sets (TE, TW, EWP), and matches the tuple members of these sets based on the shared elements, i.e., entry-Point, test, and widget.

the output TP like this (line 9).(文字)

In total, for an app consisting of T tests and P permissions, the number of test-runs by PATDroid are calculated as follows: (这个公式)

17-18

PATDroid is realized by Java code and Python script.

And PATDroid runs in two modes: Developers mode, andTesters mode.

19-20

Our evaluation of PATDroid addresses the following questions:

RQ1. Efficiency:

RQ2. Coverage

RQ3. Effectiveness:

RQ4. Performance:

21

This evaluation uses the Google play and GitHub repositories to crawl Android apps on a conditional basis. A total of 100 Google play Android applications and 10 open-source apps from Github are available.

*The paper have compared PATDroid against three alternative strategies:Exhaustive、Pairwise、All-and-None*

22

For RQ1. The paper compares the PATDroid test run time and test execution time with three alternative strategies. The paper plots the test execution time for all 110 theme applications. The results in confirm that PATDroid can significantly reduce the number of test-runs and test-execution time.

23

For RQ2. PATDroid achieves the same exact coverage as exhaustive in all subject apps. In summary, the results of RQ1 and RQ2 confirm that PATDroid is able to significantly reduce the number of tests without trading-off code coverage.

24

For RQ3. Running PATDroid on the set of 110 apps, the writer found 14 apps (i.e.,13%) with defects that are due to inappropriate handling of dynamic permissions and most of the defects are verified and fixed by the app developers. further demonstrating the efficacy of PATDroid in revealing permission-related defects.

25

For RQ4. On average, static and dynamic analyses take 97 and 259 seconds, the static analysis time increases as the app size increases, while there is no correlation between the dynamic analysis time and the app size.Dynamic analysis time depends on the logic and workload of the subject app.

26-27

PATDroid leverages a hybrid program analysis approach, by excluding the permissions that do not interact with tests, so that it can achieve a significant reduction in testing effort, yet achieve comparable coverage and fault detection capability as exhaustive testing.

The paper plans to extend the approach to include other

configurable parameters in Android that can affect the behavior of programs, such as the settings for network and battery usage.