1. Getting Started

In this link, you can download all the files needed by the artifact, or you can download these files separately in the corresponding location of README.

1.1 Running SetDroid via Virtual Machine

Requirements

- You need to enable virtualization technology in your computer's BIOS, see this link for how to enable virtualization technology in the computer. Some computers have turned on virtualization by default.
- Your computer needs at least 16G of memory, and at least 40G of storage.
- VirtualBox: we built our artifact by using version 6.1.20.
- Download the zip file (SetDroid_VM.zip) from this link (SHA256: ec274b5c23257ad1b94bc3733076b0092fd199b0b8dab5a74d852e5dfa659bf2). and extract it.

Setting up (video tutorial)

- Open VirtualBox, click "File", click "Import Appliance", then select the file named "SetDroid.ova" from the extracted contents (this step will take about five to ten minutes to complete).
- After the import is completed, you should see "vm" as one of the listed VMs in your VirtualBox.
- Click "Settings", click "System", click "Processor", and check "Enable Nested VT-x/AMD-V"
- Run the virtual machine. The username and the password are both "setdroid".

Run (video tutorial)

- Open the terminal and execute the following command:

```
/home/setdroid/Android/Sdk/emulator/emulator -avd Android8.0 -read-only -port
5554 &
```

- Wait for the first Android emulator to start. After the emulator is successfully started, return to the command-line interface, press enter, and then execute the following command:

```
/home/setdroid/Android/Sdk/emulator/emulator -avd Android8.0 -read-only -port
5556 &
```

- Wait for the second Android emulator to start. After the emulator is successfully started, return to the command-line interface, press enter, and then execute the following command:

```
cd /home/setdroid/SetDroid/Tool
```

- Then execute the following command (this step will take about five to ten minutes to complete):

```
python3 start.py -app_path /home/setdroid/SetDroid/App/a2dp.Vol.apk -
append_device emulator-5554 -append_device emulator-5556 -android_system
emulator8 -append_strategy display_immediate_1 -testcase_count 1 -choice 0 -
event_num 50
```

- At this point, SetDroid will start to run a round of example policy (Oracle checking rule I -immediate -display -1) on the example app (A2DP Volume), which contains 50 events.
- The target app can be modified by the configuration parameter `-app_path` . The number of runs can be modified by the configuration parameter `-testcase_count` . The number of events contained in each test can be modified by the configuration parameter `-event_num` . Setting change strategy can be changed through the configuration parameter `-append_strategy` . You can also add more strategies to make them be executed in sequence.
- For example, the following command represents the sequential execution of two strategies (Oracle checking rule I - lazy - permission) and (Oracle checking rule II - language) on Amaze. Each strategy is executed 10 times, and each test contains 100 events (this command will take about one to two hours to complete, and you can interrupt the command through `Ctrl-C` at any time):

```
python3 start.py -app_path
/home/setdroid/SetDroid/App/com.amaze.filemanager.apk -append_device
emulator-5554 -append_device emulator-5556 -android_system emulator8 -
append_strategy permssion_lazy_1 -append_strategy language -testcase_count 10
-event_num 100
```

1.2 Building and Running SetDroid From Scratch

Requirements

- Download all the files from this link
- Download all the apps from this link
- Android SDK: API 26+
- Python 3.8
- We use some libraries (uiautomator2, androguard, cv2, langid, numpy) provided by python, you can add them as prompted, for example:

```
pip3 install langid
```

Setting up

You can create an emulator before running SetDroid. See this link for how to create avd using avdmanager. The following sample command will help you create an emulator, which will help you to start using SetDroid quickly:

```
sdkmanager "system-images;android-26;google_apis;x86"
avdmanager create avd --force --name Android8.0 --package 'system-images;android-
26;google_apis;x86' --abi google_apis/x86 --sdcard 512M --device "pixel_xl"
```

Next, you can start two identical emulators and assign their port numbers with the following commands:

```
emulator -avd Android8.0 -read-only -port 5554
emulator -avd Android8.0 -read-only -port 5556
```

Run

If you have downloaded our project and configured the environment, you only need to enter `download_path/tool` to execute our sample app with the following command:

```
python3 start.py -app_path /home/setdroid/SetDroid/App/a2dp.Vol.apk -append_device
emulator-5554 -append_device emulator-5556 -android_system emulator8 -
append_strategy display_immediate_1 -testcase_count 1
```

SetDroid provides several ways to test android apps by command lines. You need to view configuration help through the following commands and change them.

```
python3 start.py --help
```

2 Detailed Description

2.1 All Optional Parameters of SetDroid

- `-pro_click` The proportion of click events, which is 45% by default.
- `-pro_longclick` The proportion of long-press events, which is 25% by default.
- `-pro_scroll` The proportion of scroll events, which is 5% by default.
- `-pro_home` The proportion of click home button events, which is 0% by default.
- `-pro_edit` The proportion of edit events, which is 15% by default.
- `-pro_naturalscreen` The proportion of rotating to natural events, which is 1% by default.
- `-pro_leftscreen` The proportion of rotating to left events, which is 8% by default.
- `-pro_back` The proportion of click back button events, which is 1% by default.
- `-pro_splitscreen` The proportion of split-screen events, which is 0% by default.
- `-app_path` the APK address of the app you want to test.
- `-append_device` The serial numbers of devices used in the test, which can be obtained by executing "adb devices" in the terminal.
- `-android_system` The Android system of the test device, At present, only Android 8.0 system is supported.
- `-root_path` The storage path of the output file.
- `-resource_path` The path of the resource file that you want to import into the test devices in advance.
- `-testcase_count` The number of rounds that you want to test for each strategy.
- `-event_num` The number of events in per round of test.
- `-setting_random_denominator` Used to adjust the frequency of setting change event insertion.
- `-append_strategy` The strategies that you want to use (multiple test strategies can be executed in sequence), currently, the supported strategies are as follows, corresponding to the 14 strategies listed in Table 5 of the paper.

Strategy name	Setting	Oracle rule	Injection strategy	Pair of events for setting changes
network_immediate_1	Network	I	Immediate	⟨turn on airplane, turn off airplane⟩
network_lazy_1	Network	I	Lazy	⟨turn on airplane, turn off airplane⟩
network_lazy_2	Network	I	Lazy	⟨switch to mobile data, switch to Wi-Fi⟩
location_lazy_1	Location	I	Lazy	⟨turn off location, turn on location⟩

Strategy name	Setting	Oracle rule	Injection strategy	Pair of events for setting changes
location_lazy_2	Location	I	Lazy	⟨switch to "device only", switch to "high accuracy"⟩
sound_lazy_1	Sound	I	Lazy	⟨turn on "do not disturb", turn off "do not disturb"⟩
battery_immediate_1	Battery	I	Immediate	⟨turn on the power saving mode, add the app into the whitelist⟩
battery_lazy_1	Battery	I	Lazy	⟨turn on the power saving mode, turn off the power saving mode⟩
display_immediate_1	Display	I	Immediate	⟨switch to landscape, switch to portrait⟩
display_immediate_2	Display	I	Immediate	⟨turn on multi-window, turn off multi-window⟩
permssion_lazy_1	Permission	I	Lazy	⟨turn off permission, turn on permission⟩
developer_lazy_1	Developer	I	Lazy	⟨turn on "Don't keep activities", turn off "Don't keep activities"⟩
language	Language	II	-	⟨change system language, -⟩
time	Time	II	-	⟨change hour format, -⟩

2.2 Description of Output Files (video tutorial)

- The output path of the tool is in `/home/setdroid/SetDroid/Root`.
- The result files of each app are classified and stored in `/home/setdroid/SetDroid/Root`.
- Open the folder of an app, and you will see the result files of each strategy for this app are stored by category.
- Open the folder corresponding to a strategy, and you will see an `error_realtime.txt` file, a `wrong_realtime.txt` file, and many numbered folders correspond to each round of test results.
- Open a numbered folder, and you can see a `read_trace.txt` file, a `trace.txt` file, an `i_trace.html` file, and a folder named `screen`.
- Open the `screen` folder, and you can see the screenshot of each step and the corresponding interface layout information file.
- Next, I will introduce the content and use of each file.

error_realtime.txt

This file records the sequences that trigger the setting defects, which start with `Start::x::run_count::y` (x means the x-th error and Y means the error was captured during the y-th round of execution), and end with `End::`

wrong_realtime.txt

This file records the sequences that trigger the suspected setting defects.

read_trace.txt

This file records the execution sequence of SetDroid, which is easy for SetDroid users to read.

trace.txt

This file records the execution sequence of SetDroid, which can be read and replayed by SetDroid.

i_trace.html

This file records the sequence of screenshots after each step, which is arranged horizontally. The events executed at each step are marked on the screenshot. After opening the file in the browser, there is a drag bar at the bottom, which can drag horizontally to view the whole sequence. When the error is captured, the screenshot is marked with a red frame. When the two interfaces are different, the screen capture is marked with a yellow frame.

2.3 Study Dataset

Download the zip file (Study.zip) from this link (SHA256: 6cc5514628bfef175f31c4c0a2d8a40e57af21e018441fdb3e3ce4ec12726f56), and extract it.

project_lists.xlsx

This file contains all the Android projects officially released on Google Play and F-Droid that we obtained through GitHub's REST API. We listed their repo names, the number of closed issues, the number of open issues, and the number of stars on GitHub. We keep the apps with more than 200 issues in this list as study objects, a total of 180 apps.

app_infomation.xlsx (corresponding to Figure 2 in our paper)

This file contains all the information (star numbers, issue numbers, installations, and categories) of our study objects, corresponding to the data in Figure 2 in the paper.

setting_keyword_issues.xlsx

This file contains all issues of 180 apps obtained through keyword filtering. We listed their repo name, URL, open time, closed time, and setting keywords mentioned in the issue.

study_list.xlsx (corresponding to Table 3 in our paper)

This file contains 1,074 setting issues of 180 apps that we obtained through manual inspection. For each issue, we listed its URL, consequence, whether it was closed, whether it was repaired, the length of the reproduce steps reported by the reporter, the setting that caused the issue, and the root cause of the issue.

specific_APIs_list.xlsx

This file lists the specific APIs that we used when investigating the usage of settings in the apps. We listed their corresponding setting categories, the classes they belong to, and the forms used in the static analysis.

usage_of_settings.xlsx (corresponding to Table 2 in our paper)

This file lists the APIs related to the setting categories used by each app. For each app, we counted the usage of six settings. The number 1 means that at least one API corresponding to this setting category has been detected in the code of the app.

2.4 Evaluation Data

- Download the zip file (Evaluation.zip) from this link (SHA256: ecd9da64f4f8c77beb3ab40b71c9e6ed3c74d153faa1622e6d1a02a38cf03a8f), and extract it. It contains the test results of open-source apps. Due to trade secrets, bug reports and running paths of five industrial apps cannot be disclosed.

Bugs (corresponding to Table 8 and Table 6 in our paper)

- The file called `bug list.md` records the information of 42 bugs reported by SetDroid and bugs detected by PatDroid, corresponding to columns 5-8 in Table 8 and the last four columns in Table 6 in our paper

TP and FP (corresponding to the last four columns in Table 6 in our paper)

- The trace of all errors (both TP and FP) reported by SetDroid in Table 6 and the screenshot after each step are stored in the `TP and FP` folder. Note that the tool has been reconstructed in the process of running the experiment, so the format of the output file is slightly different. The total running times are shown in section 4.3.
- There are 24 folders in `TP and FP`, corresponding to 24 apps in the second column of Table 6 of the paper. Each folder is named by the package name of the app. For the corresponding relationship between package name and app name, see `app_infomation.xlsx` (in section 2.1.2).
- Under each app folder, there are two folders `TP` and `FP`, and each folder has two folders `Rule1` and `Rule2`, corresponding to the last four columns in Table 6. That is, `/FP/Rule1/` corresponds to the column FP^{I} of Table 6, `/TP/Rule1/` corresponds to the column TP^{I} of Table 6, `/FP/Rule2/` corresponds to the column FP^{II} of Table 6, `/TP/Rule2/` corresponds to the column TP^{II} of Table 6.
- For errors detected by using `Oracle checking rule I`, as context information is needed to reproduce the errors, SetDroid will record in detail the trace that triggers each error (stored in the `trace.txt` file), the screenshots and interface layout files after each step (stored in the `screen` folder).
- For errors detected by using `Oracle checking rule II`, SetDroid will simply record the screenshots and interface layout files with errors, as well as fields that are not converted correctly according to settings (recorded in `language.txt` or `time.txt`).
- For example, if you want to verify the data of the last app `Habits` in Table 6, you can open the directory `org.isoron.uhabits` in the directory `/TP and FP/`. In Table 6, the FP^{I} of `Habits` is 2, TP^{I} is 2, FP^{II} is 0 and TP^{II} is 1, so:
    - `/Evaluation/TP and FP/com.tom.alwayson/FP/rule1/` has two folders (`1_time`, and `2_time`), corresponding to two FPs caused by changing time setting.
        - For example, `/Evaluation/TP and FP/com.tom.alwayson/FP/rule1/1_time/` have a folder `screen` and a file `trace.txt`. Open folder `screen` to see the interface information of the app before executing each step in `trace.txt` on two different devices, such as `1_emulator-5554` corresponds to the interface information before the execution of the first step in device `emulator-5554`. `1_emulator-5556` corresponds to the interface information before the execution of the first step in device emulator-5556. However, as the device `emulator-5554` uses 12-hour format and the device `emulator-5556` uses 24-hour format, after the execution of step 18, the clock styles of

the device `emulator-5554` and the device `emulator-5556` are different, so step 19 can be executed on the device `emulator-5554`, but can not be executed on the `emulator-5556`, so SetDroid reports an error. After manual confirmation, we think that this is not a bug, so we classify it in FP.

- `/Evaluation/TP and FP/com.tom.alwayson/TP/rule1/` has two folders (`1_rotate`, and `2_rotate`), corresponding to two FPs caused by changing display setting (rotating screen).
- `/Evaluation/TP and FP/com.tom.alwayson/FP/rule2/` has no content.
  - For example, `/Evaluation/TP and FP/com.tom.alwayson/FP/rule1/1_rotate/` have a folder `screen` and a file `trace.txt`. Open folder `screen` to see the interface information of the app before executing each step in `trace.txt` on two different devices, such as `1_emulator-5554` corresponds to the interface information before the execution of the first step in device `emulator-5554`. `1_emulator-5556` corresponds to the interface information before the execution of the first step in device emulator-5556. As the device `emulator-5556` rotated screen after step 15, so the user's choice of time is lost. As the result, step 18 can be executed on the device `emulator-5554`, but can not be executed on the `emulator-5556`, so SetDroid reports an error. After manual confirmation, we think that this is a bug with the consequence of data loss, so we classify it in TP.
- `/Evaluation/TP and FP/ com.tom.alwayson/TP/rule2/` has a folder `language`, in which six TPs detected after changing the language setting are recorded.

2.5 Tool Extension

If someone wants to extend the artifact, they can modify it in the following position of the tool.

Add settings change strategy

Add a new setting change function in injector.py, and add calls to it to `change_setting_before_run` or `inject_setting_during_run` as needed.

Add seed test generation policy

Add a new exploration class according to `RandomPolicy` class in policy.py, and inherit the `Policy` class

Add a new check condition

Add a new check function in check.py and call it in the corresponding position in the `executor.py`