Understanding and Detecting Evolution-Induced Compatibility Issues in Android Apps (Artefact Manual)

Anonymous

December 6, 2021

Our artefact provides the source files of IctApiFinder, scripts, benchmarks, and databases that are used in the evaluation of our paper. We did not plan to open source this artefact when the paper was published in 2018. However, as more and more people from different institutions and companies requests the sources of IctApiFinder, we now decide to open source the tool at `https://github.com/DongjieHe/IctApiFinder`. Issues and Pull requests are welcome.

The results in Table 5 of our paper could only be partially reproduced now due to two reasons:

- The dependencies have undergone a major upgrade. The original dependencies like `soot-infoflow:2.0.0-SNAPSHOT`, `soot-infoflow-android:2.0.0-SNAPSHOT`, and `heros:0.0.1-SNAPSHOT`, which could be downloaded from maven repositories in 2018, are now not available anywhere. I use their most recent version provided in `soot-infoflow-cmd-2.9.0-jar-with-dependencies.jar` as an alternative. However, some APIs exit in the old version of `soot-infoflow` like `setUseExistingSootInstance` (used in line 67 in `IctApiFinder.java`) and `setEnableMergePointChecking` (used in line 135 in `IctApiFinder.java`) have been fully obsoleted and deleted in the newest version. I just comment out these lines in `IctApiFinder.java` and am not sure how will these lines affect the analysis results since I could no longer view the original implementation of these APIs. I am also not sure how the semantics change in the newest version will affect the analysis results.
- The benchmarks used in the paper are only partially available. All the 20 benchmarks in Table 5 were downloaded from Fdroid then but only 8 of them are now available.

Anyway, I open source this tool and hope it can benefit more people in the community.

1 Artifact check-list

- **Source:** Our tool, IctApiFinder, is built on top of `FlowDroid`[1].
- **Environment:** The experiments were performed on Ubuntu 18.04, which rely on Java 8+ to execute IctApiFinder, Gradle for driving scripts, and LogicBlox to read API information from Databases (I plan to directly read API information from Android.jar in the future).
- **How much disk space required?:** Approximately 3.8GB for the docker image. However, the databases (generated by Doop) containing in the image is quite disk-consuming. So once decompressed, it will requires about 15GB or more disk space.
- **Output:** The log files produced by IctApiFinder are in "data/out/" directory (which could be configured in "data/config.json") by default.
- **Experiment workflow:** see below.
- **Publicly available?:** Yes, the artefact is publicly available under GPL v3.

[1] `https://github.com/secure-software-engineering/FlowDroid`

2 Installation

The artifact is provided in form of a prebuilt Docker image with all software dependencies included. It can be downloaded from Docker Hub at `https://hub.docker.com/r/hdjay2013/ictapifinder`.

Using Docker image Install Docker if it is not installed already by following the documentation at `https://docs.docker.com/get-docker/`. You might need to follow the post installation steps (`https://docs.docker.com/engine/install/linux-postinstall/`) for managing docker as a non-root user. Then pull the Docker image and run it with the following commands:

```
$ docker pull hdjay2013/ictapifinder:latest
$ docker run -it –shm-size=10g hdjay2013/ictapifinder:latest
```

Note that "–shm-size" option is used to specify the size of shared memory as a large shared memory is required by the LogicBox engine.

3 The Artifact Directory Tree

Once you run the command above, you will be in a terminal of the container. Use `ls` command, you will see the complete directory tree of our artefact (shown below).

```
/
├── benchmarks/
├── data/
│   ├── SDK/
│   ├── out/
│   └── config.json
├── src/
├── libs/
│   └── soot-infoflow-cmd-2.9.0-jar-with-dependencies.jar
├── platforms/
├── gradle/
├── settings.gradle
├── build.gradle
├── gradlew .2 Artifact-Manual.pdf
├── paper.pdf
└── LICENSE
```

- `benchmarks/` contains the 8 out of 20 benchmarks used in the paper.
- `data/` contains the databases of Android SDK and configuration file, i.e., `config.json`.
- `src/` is source of IctApiFinder.
- `libs` contains `soot-infoflow-cmd-2.9.0-jar-with-dependencies.jar`.
- `platforms` contains different levels of Android SDK packages.
- `gradle/`, `settings.gradle`, `build.gradle`, and `gradlew` are standard files in projects managed by the Gradle tool.
- `Artifact-Manual.pdf` is this file.
- `paper.pdf` is the paper in submission.
- `LICENSE`: the artifact is released in GPL v3.

3.1 How to use IctApiFinder?

The following command analyzes a single Android Application:

```
$ ./gradlew run -Pargs="/ictapifinder/benchmarks/jonas.tool.saveForOffline_21.apk"
```

The output is in "/ictapifinder/data/out/jonas.tool.saveForOffline_21.apk.report" which is shown below:

```
jonas.tool.saveForOffline_21.apk minSdkVersion: 16, targetSdkVersion: 27
BUG: <android.webkit.WebSettings: void setMediaPlaybackRequiresUserGesture(boolean)> called in
<jonas.tool.saveForOffline.ViewActivity: void setupWebView()> on line -1 not in [16] reachable path:
--><dummyMainClass: void dummyMainMethod(java.lang.String[])>
--><dummyMainClass: jonas.tool.saveForOffline.ViewActivity dummyMainMethod_jonas_tool_saveForOffline_ViewActivity
(android.content.Intent)>
--><jonas.tool.saveForOffline.ViewActivity: void onCreate(android.os.Bundle)>
--><jonas.tool.saveForOffline.ViewActivity: void setupWebView()>
```

IctApiFinder could also work in batch mode. For example, the following command will analyze all Android applications under "/ictapifinder/benchmarks/".

```
$ ./gradlew run -Pargs="/ictapifinder/benchmarks/"
```

That's all about this artefact. Have fun!