

SnT **ATUA** **User Manual**



Prepared by Chanh Duc NGO


Copyright: University of Luxembourg, SnT Centre for Security Reliability and Trust

Version: V1.1.0

1. Requirements

- JDK 1.8+ is required for all the tools specified in the following sections.
- Python 3 is required for Gator
- Android SDK
- Android 23+ emulator or device

2. Installation

- To install ATUA please do the following
- Go to <https://dropit.uni.lu/invitations?share=7a7ff9d345a3a37422ba>
- Click on "Download folder as ZIP" 
- Uncompress the folder in your local drive
- ATUA can be run in a native environment on a local machine.
- All the tools are provided with executables.
- The ATUA distribution is organized as follows:
 - |-- appdiff: contains Appdiff's executable
 - |-- apk-instrumentation: contains the extended version of DM2 instrumentation tool.
 - |-- ATUA: contains ATUA's executable
 - |-- demoTestData: some example apps under test
 - |-- gator: contains Gator's executable
 - |-- template: templates for manual inputs and manual intents

3. Usage

3.1 Prepare the App under test

ATUA needs the instrumented apk of the App under test, with a json providing instrumentation information, and the EWTG App model file. In addition, users can provide manual inputs and manual intents (a template for these inputs is provided in the **template** folder).

Put the necessary files of the App under test into the folder *Exploration.apksDir*. The distribution comes with some case study Apps. The inputs files's formats are the following:

- Instrumented apk: *[appPackage]_[v1]_[v2]-instrumented.apk*. This file is generated by an extended version of Droidmate2

- Instrumented info: `[appPackage]_[v1]_[v2].apk.json`, contains the list of instructions and methods of the app under test. This file is generated with the instrumented file by an extended version of Droidmate2
- EWTG App model: `[appPackage]_[v1]_[v2]-AppModel.json`. This file is generated by Gator.
- Manual input: `[appPackage]_[v1]_[v2]-input.json` . This file is expected to be filled with manual inputs to be used for testing an App.
- Manual intent: `[appPackage]_[v1]_[v2]-intent.json` . This file is expected to be filled with manual inputs to be used for testing an App intents.

3.1.1 Identify target methods and instructions

This step concerns the identification of the methods modified and introduced by an updated version. This is accomplished by executing AppDiff. This step is not necessary to run ATUA with the provided case studies.

AppDiff is provided as an executable. To run AppDiff, use the following command:

```
java -jar appdiff.jar -o diff -a <android_sdk_jar> -j
<output_folder> <path_to_oldVersionApk> <path_to_newVersionApk>
```

A diff file in json format is produced and put in the `<output_folder>`

3.1.2 Generate the EWTG App model

This step concerns the generation of EWTG App model using an extended version of Gator.

Before executing gator, put the app under test and the diff file in the same folder. For example: **gator/apk**. If the diff file is not provided, the app under test will be analysed as all the methods are updated.

To execute gator, use the following command from the **gator** folder:

```
./gator a -p <Apk Path> -outputFile <EWTG App Model path>
-worker 4 -appPackage <app package of the App under test> --timeout 36000
```

- **worker**: the number of processors which will be used
- **appPackage**: the source code's package name. Some applications have their declared package name which does not match their source code's package name. This parameter allows gator to analyse these kinds of apps.
- **timeout**: in second. The default timeout of Gator is 1 hour. This parameter could help to deal with large apps which require long processing time.

3.2.3 Instrument the app under test

This step concerns the instrumentation of the app under test using an extended version of DM2 instrumentation project.

- Firstly, put the apk file of the app under test and the diff file into the folder **apk-instrumentation/apk**. If the diff file is not provided, the app under test will be instrumented as all the methods are updated.
- Run instrumentation from **apk-instrumentation** folder with the following command

```
bin/coverage --useAppt2=<false> --packageName=<codePackage>
```

- **useAppt2**: some apps require appt2 to decompile and compile
- **packageName**: the source code's package name. Some applications have their declared package name which does not match their source code's package name. This parameter allows the tool to instrument these kinds of apps.
- The output of this step is contained in the **instrumentedApsks** folder, including the instrumented apk and the instrumented information.

3.3 Run testing with ATUA

3.3.1 Prepare the testing configuration

1. Reuse defaultConfiguration-ATUA.property file or duplicate it for new purpose

2. In the configuration file, consider the following properties:

- **Output.outputDir** : the relative path of the output folder of the tool. If the folder does not exist, the tool will create the new folder. *Attention: everytime ATUA is started, this folder will be cleaned.*

ex: Output.outputDir=./out60m/ATUA

- **Exploration.apksDir**: the relative path of the folder where the app under tests are located ex: Exploration.apksDir=./apk
- **Exploration.deviceSerialNumber**: the serial number of the device (got from "adb devices" command). *Attention: please make sure that the device is available before executing the tool.*

ex: Exploration.deviceSerialNumber=emulator-5554

- **Selectors.timeLimit**: testing time budget in minutes ex: Selectors.timeLimit=60
- **Selectors.actionLimit**: this is useful if you prefer the maximum number of actions as the overall budget ex: Selectors.actionLimit=1000
- **RegressionStrategy.budgetScale**: scaleFactor to be used for each phase. The default value should be 1.0 for 1 hour testing budget. For five hours, we suggest using 2.0.

3.3.2 Execute ATUA

ATUA can be run from the command line. You can specify the configuration file to be used with the parameter "--Core-configPath". The following command is run under the **atua** folder:

```
bin/atua --Core-configPath=./defaultConfig-ATUA.properties
```

3.3.3 Inspecting results

The result is stored in the **[outputDir]/droidmate** :

- **coverage/**: contains details of coverage of each action (generated by DM2)
- **report/**: contains graph visualization of the entire execution (generated by DM2)
- **logcat.log**:
- **model/[appPackage]**: contains the model and the coverage related to updated methods.
 - **atua-report.txt**: contains the overall result in which you can find:
 - The number of statements (instructions)
 - The number of methods
 - The number of updated methods
 - The number of statements belong to the updated methods
 - The coverage
 - Some additional information including:
 - List of covered updated methods
 - List of uncovered update method
 - The coverage ratio after each phase
 - Unreachable windows
 - **coverage.txt**: contains the list of instructions with the first covered timestamp.
 - **crashlist.txt**: contains the list of crash exceptions (generated by DM2)
 - **EWTG/**: contains the enhanced EWTG's part
 - **DSTG/**: contains the DSTG generated by ATUA
 - **States/**: contains the captured GUITrees.
 - **trace<id>.csv**: contains all the inputs generated and transitions after each action.
 - **Images/**: contains the screenshots of apps. ATUA takes a screenshot after every input being triggered.
 - **actionCoverage.csv**: contains the summary of coverage of each action.