

Replicable Testing of Android Applications with Replaykit

Dan Watson
daniel.watson.3141@gmail.com
University of Waterloo, SWAG

June 27, 2018

1 Setup

Note: In some cases, commands are too long to fit nicely into this document, so they have been split across a few lines. All such commands given in a single step are one single command and will only work as such.

1.1 Android Debugging Bridge

1. Make sure adb is working. Type:

```
adb
```

It should give you version information and instructions. If it does, the SDK and ADB are working, and you can go to step 4.

2. Install Android SDK cli tools.
3. Ensure `<sdk>/platform-tools` is in your PATH.
4. Check the ADB again to ensure everything is installed properly.
5. Enable USB debugging on the Android device.
6. Enable "show touches" in the developer options as well. This will help us later.
7. Plug your phone into your computer with an MTP (Media Transfer Protocol) capable cable.
8. Ensure your phone has USB MTP enabled in developer options. You can also check if the phone is seen as a storage device by your OS. If so, your phone is in MTP mode.
9. Test the link by sending a command to the phone over the ADB. With the screen off, type:

```
adb shell input keyevent KEYCODE_WAKEUP
```

Your phone should activate.

1.2 Trepn Profiler

1. Install Trepn Profiler, available on Google Play Store.
2. We need to configure Trepn to record what we want to measure. Open trepn and go to settings>Data Points.
3. Select which data-points to collect. I recommend CPU Load(Normalized), GPU Load, Memory Usage, and Screen State. Screen state is to help with processing (more on that later). The others are relevant performance data-points.
4. Be sure to save your preferences. Open Settings>general>Save Preferences. Remember what you name this file, save it to a variable in your console, or write it into a text file so you can copy and paste it later.
5. We also need to locate the install directory for Trepn. Open your phone's file browser, or browse your phone from your computer and locate the "trepn" folder, usually in the root directory unless it was installed to an SD card. Get the path of this directory (relative to the phone, not your computer) and save it to a variable or write it down somewhere. Example: "root/trepn" or "sdcard/trepn".

1.3 ReplayKit

For more information, see: [Replaykit's Github](#)

1. Find the executable file called starfish.exe or appetizer.sh
2. From shell terminal in Portable/replaykit-master/<Your OS>, get the path to this executable and add it to your PATH.
3. For Mac and Linux, be sure to set the .sh file to executable. This is usually (but not always) unnecessary on Windows.
4. Test your installation. Type:

```
cd ..  
starfish #or  
appetizer
```

depending on the name of the executable. It is starfish for Windows and appetizer for Linux and Mac. It should tell you "incorrect usage" as opposed to "command not found".

5. Now find your phone's uid, usually the same as the serial number. This is how we will identify the phone for our experiments. Type:

```
starfish devices list
```

You should see a list of attached Android devices. The first item in the list will be marked "uid". This is the field we need. Save it to a variable or copy it down (with quotes). Example:

```
$myPhone = "abcd1234"
```

2 Basic Command-Line Functionality

This section is for reference and you need not type these commands into your terminal if you are just getting set up.

2.1 Android Debugging Bridge

We can send commands to the phone over the ADB. With your phone screen off, type:

```
adb shell input keyevent KEYCODE_HOME
```

Your phone should activate after a moment. If it does not, either the adb is mis-configured, or the phone is not connected properly. You can find lots of helpful adb commands [HERE](#)

2.2 Trepro Profiler

We can tell control trepro using the Trepro Profiler app.

1. You can load your preferences file in the settings menu. Using our \$treproPath and \$prefsFile variables from before, we can also load our preferences in this way:

```
adb shell am broadcast
-a com.quicinc.trepro.load_preferences
-e com.quicinc.trepro.load_preferences_file
-join($treproPath ,"/saved_preferences/", $prefsFile)
```

2. Start Profiling:

```
adb shell am broadcast
-a com.quicinc.trepro.start_profiling
-e com.quicinc.trepro.database_file "treproTemp"
```

3. Stop Profiling:

```
adb shell am broadcast
-a com.quicinc.trepro.stop_profiling
```

4. Convert output db file to csv

```
adb shell am broadcast
-a com.quicinc.trepro.export_to_csv
-e com.quicinc.trepro.export_db_input_file "treproTemp.db"
-e com.quicinc.trepro.export_csv_output_file $outputFile
```

2.3 Replaykit

Any command we can send over the adb, we can send over starfish. For example:

```
starfish devices control $myPhone shell input keyevent KEYCODE_WAKEUP
```

2.3.1 Testing Multiple Devices

One thing Replaykit allows us to do easily that the adb does not is control multiple devices at once. This functionality is optional, but allows you to control multiple phones if you wish, and is fully compatible with the rest of the testing technique presented here.

1. We first create an array of uid's:

```
$myPhones = $phone1,$phone2,$phone3
```

2. Then wake them all up:

```
starfish devices control $myPhones  
shell input keyevent KEYCODE_WAKEUP
```

We can always use our array of phone id's instead of a single id in our scripts.

2.3.2 Recording touch interactions

More detail can be found on the GitHub.

Start recording:

```
starfish trace record --device $myPhone mytrace.trace
```

type 'exit' to stop.

Check the trace (optional): `starfish trace info mytrace.trace`

This gives information about the trace such as duration and number of recorded events. If numbers seem off, something isn't working right.

We can also view and edit the trace file by decompressing it, then opening it with a text editor. Replaykit is ok with compressed or decompressed trace files but they are compressed by default

Play back the Trace:

```
Starfish trace replay mytrace.trace $myPhone
```

I recommend using CPU Load(Normalized), GPU Load, Memory Usage, and Screen State

Screen state is to help with processing (more on that later). The others are relevant performance datapoints. Be sure to save your preferences by hitting general;Save Preferences. Remember what you name this file (or save it to a variable in your console)