Testing Soot with Dex Support

Authors: Michael Markert, Frank Hartmann 09.05.2012

Purpose of this document

Assuming the reader is known to the Soot Project, this document gives an insight into the scope of the testing process, the actual testing scenario and the applications that were tested with our dex plugin of Soot. For further information, please contact the authors of the Soot Project at http://www.sable.mcgill.ca/soot/

Scope

The scope of the testing process is limited to some sample Android applications which differ in complexity, as well as an application that exposes all available byte codes that are subject to test. All listed applications were undertaken the following testing process.

Testing Process

Every application is tested along the following steps to ensure a valid transformation of dex files with Soot in combination with our dex plugin. Android applications which were transformed with our plugin should still show the same behavior as if they were not touched by Soot.

Therefore, the following process illustrates how Soot transforms a runnable Android application into java class files which are compiled back into the same runnable Android application in return.

1.) Compile dex file and install the android application on the android emulator. The installed application should run properly.

2.1) convert the compiled application with soot into its corresponding class files.
2.2) optionally, compile the application into its jimple representation and verify manually.

java -jar soot.jar -src-prec dex -cp <full-path-to-android.jar>;<full-path-to-andoird-application-binaries> -app <url-of-application> -output-dir <some-output-dir> -output-format <class/jimple> -x android

For example:
<full-path-to-android.jar> = C:\Users\Frank\git\soot-dalvik\lib\android.jar
<full-path-to-andoird-application-binaries> = C:\Users\Frank\workspace\Snake\bin\
<url-of-application> = com.example.android.snake.Snake
<some-output-dir> = C:\Users\Frank\git\soot-dalvik\sootified\
<class/jimple> = class

Class files are then generated and stored in the chosen output directory.

3.) Translate the class files back to a single dex file by using the android tool dx as follows.

dx --dex --no-strict --output=<path-to-output-dex-file> < folder-of-class-files >
For example:
<path-to-output-dex-file>= C:\Users\Frank\smali\classes.dex
<folder-of-class-files> = C:\Users\Frank\git\soot-dalvik\sootified\com\example\android\snake

4.) Replace the newly created classes.dex with the old dex file of the archive APK of the Application

5.) Re-sign the archive APK, that contains the new classes.dex
For example:
java -jar signapk.jar testkey.x509.pem testkey.pk8 <input-apk> <output-apk>

6.) Deinstall the old application from the android emulator and install the new application with the android tool adb onto the emulator.
For example:
adb uninstall com.example.android.snake
adb install Snake_new.apk

7.) Start the application on the emulator and everything should behave as in 1.)

Tested Applications

The Android applications that were tested originate from Androids example repository.

com.example.android.snake.Snake
com.example.android.livecubes.cube1
com.example.android.livecubes.cube2
com.example.android.jetboy
com.example.android.notepad
com.example.android.BluetoothChat
com.android.example.spinner

Additionally, we created an example application, which covers all available byte codes that can be handled by the Dalvik VM.
soot.dex.exampleApp

Results

All applications were successfully undertaken the testing process that is described above.