Source Code Documentation for STIGMA (all .py files)

**Stigma.py**

Main program that carries out APK modification and introduces code to enable tracking of leakage of sensitive information. Also carries out object extraction on the subject apps, where the objects can then be used for various specific purposes upon addition of plugins.

**ControlFlowGraph.py**

This class builds a control flow graph for a given method text using the networkX library. This graph generates nodes for each block of code (list of all contingunous instructions) until it sees an if, cond, goto, switch, or return, in which case it branches out, creating a new node. Then it creates an edge from existing blocks of code to this new branching node.

https://networkx.org/documentation/stable/index.html

**Instrumenter.py**

API for plugins. 2 primary methods: sign\_up and sign\_up\_method\_start

**JSONTrailPlugin.py**

Plugin for “Object Extraction”

**SafeRegisterCollection.py**

Program that collects safe registers, which are:

1. Low numbered (less than v16)
2. Not containing original program data
3. Not the second ½ of a wide value
4. Not used in the code\_unit

**SimpleTaintTrackingPlugin.py**

Prototype of experimental idea. Unfinished

**sink\_parser.py**

Parser method to look up from SourcesAndSinks.txt file, extract all Sinks and write in the Sink.txt file

**SmaliClassDef.py**

Python object to represent smali class definitions

**SmaliMethodDef.py**

Python object to represent smali method definitions. Many are contained in SmaliClassDef.py. Has 2 classes: SmaliMethodDef and SmaliMethodSignature

**SmaliAssemblyInstructions.py**

Stores python objects that correspond to every smali instruction

**SmaliRegister.py**

Class represents the smali registers.

**SmaliTypes.py**

Object to represent smali type system

**SmaliCodeIterator.py**

Most instructions are 1 line long, but some exceed this number. File helps iterate through all instructions smoothly.

**StigmaStringParsingLab.py**

Program to parse through Smali registers. Returns a list of strings of all those required after parsing

**StigmaTests.py**

Automated testing

**TaintStorageHandler.py**

Stores the taint tags using a specific naming convention

**TaintTrackingInstrumentationPlugin.py**

Plugin for “Information Leakage”. Enables Stigma to inform users when sensitive information (GPS location, IMEI number, phone number) is leaked across a network.

**TypeSafetyChecker.py**

Analyzes and identifies data types in every register at every point in the app.

**ValidSmaliInstruction.py**

Collection of Smali Instructions to pass on smali files of APK being modified. Called upon by various other .py files, who themselves do the modifications.

**gplaycli/get\_them\_all,py**

Auxiliary program that helps download apps from the Google Play Store. DEPRECATED!

**app\_check\_eval.py**

Auxiliary file. Used to generate figures for evaluation.

**cpu\_overhead\_eval.py**

Auxiliary file. Helps evaluate the CPU overhead generated by running the tracked APK. Plots a graph showcasing the CPU overhead

**implicit\_flow\_eval\_prelim.py**

Auxiliary file. Evaluates the number of implicit flows identified on 3 tested apps. These logs were counted manually. Plots a graph showing these identifications

**loc\_eval.py**

Auxiliary file. Program that helps evaluate the addition in Lines of Code after Stigma has successfully modified an APK. Generates a graph that suggests the comparison between LOCs before running Stigma and after.

**JavaVerifierCrashHelper.py**

Auxiliary program

**LogLeaks.py**

Auxiliary file. Program overseeing Logcat Leak Detection Protocol. Observes any data leaked out of sink or leaked from source.

**reg-freq-counter.py**

Auxiliary file. Measures the frequency of each register’s usage.

**SmaliChangesInvestigator.py**

Auxiliary file. Debugging platform