1 Infrastructure

Here are the tools that we used to produce the prototype and the demo videos.

Android Phone The phone that we use for the experimentation is Samsung Galaxy 4, model number of GT-I9506, and Android version 5.0.1

Android Studio The tool we used to develop the prototype of an application that can do the type checking and a test app. You can get it from http://developer.android.com/develop/index.html.

OCaml The tool we used to develop the prototype of certificate translation. You can get it from http://ocaml.org

Vysor A google chrome app to show the screen of an Android phone on the computer screen.

2 Instructions for Constructing Certificate

At this stage, since we only have the translation from JVM classes to DEX, we need to get the classes from a half compiled Android app. Using the Android Studio we can do this by first building the non-optimized app, then get the classes from the intermediates directory. After we get the classes we feed these classes into the OCaml program to get the certificate which contains the typing for the Android app. Finally we rename the certificate into "Certificate.cert" and insert it into the "assets/" directory, and compile the application.

3 Instructions for Compilation

3.1 APK Reader

We developed this APK Reader using the Android Studio tool, which you can get from http://developer.android.com/develop/index.html. We have tested this app in our own commercial Android phone.

3.2 Test application

We developed this test using the Android Studio tool. A note about this test application is that we have to put the option of no-optimize for the DX compiler. This is a little bit complicated with the Android studio which uses gradle version 1.5. One way to circumvent this problem is to build the app using lower version of gradle (we used 1.3.1). To do this we can modify the graddle setting for the project (contained in "build.gradle" at the root of the project directory), changing the value of classpath from "com.android.tools.build:gradle:1.5.0" to "com.android.tools.build:gradle:1.3.1". Then we modify the graddle setting for the module (contained in "app/build.gradle") . We inject the following code into the gradle file:

```
afterEvaluate {
    tasks.matching { it.name.startsWith('dex') }.each { dx →
    if (dx.additionalParameters == null) { dx.additionalParameters = [] }
    dx.additionalParameters += '–no-optimize' }
}
```

We have tested this app in our own commercial Android phone.

3.3 Certificate Translation

There is a makefile for the OCaml program already, so we just need to type "make" in in the root directory. The output will be a binary called "dx_ocaml".

4 Instructions for Type Checking

After opening the application, just provide the package name for the application that you want to type check into the input text, and then click on the button. The application will then parse the DEX file (which takes a really long time) and the certificate, then provide the final judgement whether the application is type check or not.

5 File Contents

Apart from this file, the contents of this repositories are:

Extended Paper.pdf Contains the extended contents for the paper

files/simpleapp.apk A simple test app which contains no method calling and exception handling in the program. This app is compiled using the option of no-optimize.

files/apkreader.apk A prototype of the proof of concept. This app can read other app's APK and then check it against the certificate. The checked app can not contain method calling and exception handling. The app must also be compiled with the flag no-optimize.

files/SimpleApp/ The source for the test app.

files/APKReader/ The source code for the prototype. Two main contents of this app are the component to parse another file's apk, and the component to type check it.

files/dx_ocaml/ The source code for the OCaml files to provide a certificate for an app given its classes source. This component will first parse the JVM class, do a naive JVM type inference, then translate the certificate.

files/Certificates/ Some sample certificates that match the test app. There are several certificates there which consists of a proper certificate, and other modified certificates which will trigger the type checking to fail. There are also files which highlight why the type checking will fail.

Certificate.cert This one is a proper certificate which will pass the type checker.

Certificate (Const rt).cert This certificate will trigger a failure in the type checking because the registers typing for the Const instruction is $\notin$ its successor's.

ConstRT_typing Show the typing for Const instruction and its successor's

Certificate (Invoke Constraint).cert This certificate will trigger a failure in the type checking because the registers typing for the Invoke instruction is not adequate. In particular, the return value of the invoked method is H while successor's res is L.

InvokeConstraint_typing Show the policy of the method invoked, and the typing for Invoke instruction and its successor's.

Certificate (mismatch instructions).cert This certificate will trigger a failure in the type checking because the number of instructions in the actual DEX file and the certificate does not match.

Certificate (Move Constraint) This certificate will trigger a failure in the type checking because the Move instruction is trying to move a value with high security level to the local variable which has low security level.

MoveConstraint_typing Show the policy of the method and also the typing for the Move instruction.

Certificate (MoveResult rt).cert This certificate will trigger a failure in the type checking because the registers typing for the MoveResult instruction $\notin$ its successor's.

MoveResultRT_typing Show the typing for the MoveResult instruction and its successor

Certificate (Move rt).cert This certificate will trigger a failure in the type checking because the registers typing for the Move. instruction $\notin$ its successor's.

MoveRT_typing Show the typing for the Move instruction and its successor

Certificate (Nop rt).cert This certificate will trigger a failure in the type checking because the registers typing for the CheckCast. instruction $\notin$ its successor's. We label the instruction as Nop in this prototype because we have not fully implement the exception handling mechanism, and as such we just treat CheckCast as another Nop instruction.

NopRT_typing Show the typing for the CheckCast instruction and its successor

files/Demo/ This folder contains the screen capture of the modification needed to enable the flag no-optimize. This folder also contain the screen recordings of the running examples of the certificates and how to translate the certificate from JVM to DEX. As a note, these videos have been edited to shorten the idle waiting.

DemoApp-TypeCheck.mp4 This is a screen recording of a successful type checking on the actual Android phone.

DemoCert-ConstRT.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (Const rt).cert"

DemoCert-InvokeConstraint.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (Invoke Constraint).cert"

DemoCert-MoveConstraint.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (Move Constraint).cert"

DemoCert-MoveResultRT.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (MoveResult rt).cert"

DemoCert-MoveRT.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (Move rt).cert"

DemoCert-NopRT.mp4 This is a screen recording of a failed type checking corresponding to the file "Certificate (Nop rt).cert"

DemoCert-Setup.mp4 This is a screen recording of how to get the intermediate Java classes, then feed it along with the policies and CDR for the JVM bytecode to the OCaml program to produce the DEX certificate. The produced certificate is then bundled along with the test application.

Gradle1.png This is a screen capture of modifying the gradle version in the Android Studio. We need to change the highlighted value into "com.android.tools.build:gradle:1.3.1"

Gradle2.png This is a screen capture of modifying the gradle setting of the app to enable the compilation with flag no-optimize.