Android cross-platform support study Summary

Name: Xinwen Li

About this Summary

This summary is an general study summary covering reverse enginerring issues with Android applications developed by cross-platform enviroment.

How does current popular cross-platform platform works?

- Xamarin.Android Architecture and Mono: Xmarin.Android is Micriosoft cross-platform developing enevironment support C# based on .Net framwork. The code is compiled into managed assembly (.dll file) and running within Mono CLR. Native code is called by Java Native Interface(JNI).
- React and React Native: JavaScript framwork developed by facebook. The JavaScript code will be transfered by syntax transformer into a bundle file and then loaded into JavaScript interpreter as bridge.
- Flutter: An open-sourced app SDK developed by Google for building cross-platform application support Dart language. Dart source code are compiled by Flutter's toolchain as bytecode. Then this byte code is loaded into Flutter kernel. Native code is called by Java Native Interface(JNI).
- PhoneGap Cordova: HTML and JavaScript cross-platfrom platfrom from Adobe. HTML and Javascript code is interpreted and loaded by abstract bridge inside Cordova container.

For more information, please see:
https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/jniTOC.html
https://docs.microsoft.com/en-us/xamarin/android/internals/architecture
https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html
https://flutter.dev/docs/resources/inside-flutter
https://dart.dev/guides/language
https://cordova.apache.org/docs/en/latest/guide/overview/index.html#architecture

How much Android applications use cross-platform supprt?

We scaned upto 10,624 Android applications and discovered 1070 apps (about 10%) developed by 5 polupar cross-platfrom enviroment. In this 1070 applications, we find 136 Mono apps, 451 React apps, 81 Flutter apps and 409 Cordova apps.

Note Obfuscation tools does not effect our scan result becaused all the requried classes name cannot be obfuscated based on our test result.

For detailed implementation, scan result and the method of identification, please see:
https://bitbucket.org/purseclab/androidnativelibidentifier/src/master/Code
https://bitbucket.org/purseclab/androidnativelibidentifier/src/master/APKDBScanResult/apks_1221_10649_no_game

About AndroidX

AndroidX is the package name of all libraries within Jetpack. Android Jetpack is a suite of libraries to helps developers follow best practices, reduce boilerplate code, and write code that works consistently across Android versions and devices so that developers can focus on the code they care about. . AndroidX is a full replacement of the support library.
For more information, please see:
https://developer.android.com/jetpack/androidx/explorer

AndroidX and FlowDroid

FlowDroid is a tool help statically computes data flows in Android apps and Java programs. However, based on our testing, this tool does not detect function callbacks originated from AndroidX libraries. We tesed the same programming logic based on android libraries with the same sink and source rule, the only difference is AndroidX packages in use. After we checked the FlowDroid Tool we diva into the part soot-infoflow-android/src/Soot.jimple.infoflow.android.SetupApplication.java does not detect the callback functions in this.callbackMethods.values() This is a Set holds all the callbackFunctions FlowDroid detected. We suspect the function callback is filtered by Analyzers from soot.jimple.infoflow.android.callbacks.

For more information, please see:
https://github.com/securesoftwareengineering/FlowDroid
https://www.bodden.de/pubs/far+14flowdroid.pdf