Demo: Reptor: Enabling API Virtualization on Android for Platform Openness

Taeyeon Ki, Alexander Simeonov, Chang Min Park, Karthik Dantu, Steven Y. Ko, Lukasz Ziarek
Department of Computer Science and Engineering
University at Buffalo, The State University of New York
{tki, agsimeon, cpark22, kdantu, stevko, lziarek}@buffalo.edu

Abstract

We demonstrate Reptor, a bytecode instrumentation tool enabling *API virtualization* on Android. It provides a general way to alter functionality of platform APIs on Android. With Reptor, third-party developers can modify the behavior of platform APIs according to their needs. All modifications are completely at the app layer without modifying the underlying platform. This allows practical openness—third-party developers can easily distribute their modifications for a platform without the need to update the entire platform.

1. INTRODUCTION

Oftentimes, mobile platform source code is closed. For example, the source code for iOS and Windows Mobile is closed. Third parties cannot really examine and modify the source code. One exception is Android that is open to the public. Third-party developers can access and modify the source code. However, there is no easy way to deploy the platform modifications to large-scale users. This is because Google or major mobile vendors such as Samsung and LG, control the platform distribution channels. In other words, there are only a select few players who can control the innovation on Android. Thus, there is a stiff barrier to realize novel ideas to make advances. To mitigate this lack of mobile platform openness, we propose a new technique called *API virtualization* to enable open innovation in Android. API virtualization allows third-party developers to modify, reimplement, or customize the behavior of platform APIs and deploy them seamlessly.

2. API VIRTUALIZATION

API virtualization is a shim layer between the platform layer and the app layer as shown in Figure 1. We create replacement classes for all platform API classes used by an app, and modify the existing code in the app to use the replacement classes instead of the original platform API classes. This shim layer contains all replacement classes and is injected into the app. In other words, API virtualization is self-contained in the app layer allowing for seamless modifications. Third-party developers can easily modify the behavior of platform APIs by modifying the implementation of replacement classes. Moreover, third-party developers seamlessly deploy their instrumented version of app through regular app distribution channels such as Google Play. This approach has an extra benefit that per-app customizations are possible. One example is when each app is tailored to sync with public cloud or secure cloud depending on its purpose as envisioned by BlueMountain [2].

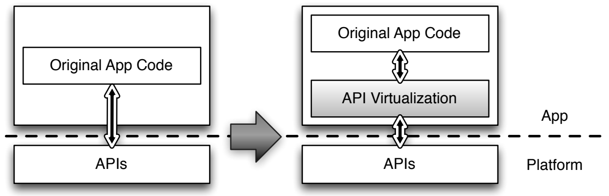

Figure 1: API Virtualization

We have implemented a prototype of API virtualization called Reptor on Android. To correctly realize API virtualization, we have carefully addressed many practical challenges. These challenges mainly come from Android's event-driven programming model and its development language, Java. Java has a variety of features such as polymorphism, which means that a child class object can be cast to a parent class anywhere in an app. Also, the child class can define its own functionality or use some of the parent class's functionality. Therefore, known techniques such as search-and-replace approach and call site instrumentation are not suitable for API virtualization due to the features.

3. DEMONSTRATION

To concretely demonstrate the power of Reptor, we will show three use cases. The first use case of Reptor is a vendor-tied library switching. In this use case, we automatically transform Airbnb app developed with Googles library to use Amazons library by using Reptor, specifically Google Maps APIs to Amazon Maps APIs. The second use case is a runtime permission mechanism that asks a user to grant permission to an app at run time, mirroring the iOS permission model. In this use case, we use Twitter as a target app. The third use case is an HTTP-to-HTTPS translator, which makes an app use HTTPS instead of HTTP if HTTPS is available on a requested URL. We have posted demo videos of the use cases on our YouTube channel [1].

4. REFERENCES

[1] Reptor Use Case Demos. https://goo.gl/hGTjpL.

[2] S. Chandrashekhara, K. Marcus, R. G. M. Subramanya, H. S. Karve, K. Dantu, and S. Y. Ko. Enabling Automated, Rich, and Versatile Data Management for Android Apps with BlueMountain. In HotStorage, 2015.

MobiSys'17 June 19-23, 2017, Niagara Falls, NY, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4928-4/17/06.
DOI: http://dx.doi.org/10.1145/3081333.3089338