Improving the Android Development Lifecycle with the VALERA Record-and-Replay Approach

Yongjian Hu Tanzirul Azim Iulian Neamtiu
University of California, Riverside
{yhu009, mazim002, neamtiu}@cs.ucr.edu

Abstract

As smartphones become more and more popular, developers are switching their focus from traditional desktop programs to mobile apps. Recording and replaying the execution of mobile apps is useful in development tasks, from reproducing bugs to profiling and testing. However, achieving effective record-and-replay on mobile devices is a balancing act between accuracy and overhead. Prior record-and-replay approaches have focused on replaying low-level instructions, which impose significant overhead. We propose a novel, stream-oriented record-and-replay approach which achieves high-accuracy and low-overhead by aiming at a sweet spot: recording and replaying sensor and network input, event schedules, and inter-app communication via intents. To demonstrate the versatility of our approach, we have constructed a tool named VALERA that supports record-and-replay on the Android platform. Through an evaluation on 50 popular Android apps, we show that: VALERA's replay fidelity far exceeds current record-and-replay approaches for Android; VALERA's low-overhead allows it to replay high-throughput, timing-sensitive apps; With the ability to deterministically replay event schedule, VALERA can help to reproduce and verify event-driven races.

Categories and Subject Descriptors D.2.4 [*Software Engineering*]: Software/Program Verification—Reliability, Validation; D.2.5 [*Software Engineering*]: Testing and Debugging—testing tools

General Terms Reliability, Verification

Keywords Mobile applications, Record-and-replay, Google Android, App testing, Event-based races

MobileDeLi'15, October 26, 2015, Pittsburgh, PA, USA
ACM. 978-1-4503-3906-3/15/10...$15.00
http://dx.doi.org/10.1145/2846661.2846670

1. Introduction

As smartphones and the applications ("apps") running on them continue to grow in popularity [5], client-side development is shifting away from traditional desktop programs and towards smartphone apps. For example, our recent study that compared bugs and bug-fixing on desktop and mobile platforms [8] has revealed that nowadays mobile bugs have higher severity than desktop bugs, and mobile bugs are fixed much faster than desktop bugs. Moreover, for Android, the study has revealed that the majority of high-severity bugs are due to concurrency errors, which creates an impetus for tools that can help find and fix Android concurrency errors.

The mobile development lifecyle includes activities such as testing, reproducing bugs, and profiling. To support these activities, we have developed VALERA (VersAtile yet Lightweight rEcord and Replay for Android) [2]. VALERA records and replays smartphone apps, by intercepting and recording input streams and events with minimal overhead and replaying them with exact timing.

While useful, record-and-replay on smartphones has proven difficult: smartphone apps revolve around concurrent streams of events that have to recorded and replayed with precise timing. To keep overhead low, prior record-and-replay approaches for smartphones only capture GUI input [4] which hurts accuracy as they cannot replay input from the network or sensors; or events, to reproduce event-based race [6]. Prior work on record-and-replay for desktop and server platforms [1, 7] has relied on techniques such as hardware changes, or instruction logging which is too heavyweight for mobile apps. To address these challenges, we introduce a novel, *sensor- and event-stream driven* approach to record-and-replay; by focusing on sensors and event streams, rather than system calls or the instruction stream, our approach is effective yet lightweight.

Our experiments shows that: VALERA is able to replay 50 popular Android apps which use a variety of sensors with low overhead (about 1% for either record or replay). VALERA is effective for reproducing bugs by replaying the input and event schedule that led to an error state. With the support of deterministic replay of event schdules, VALERA is able to reproduce hard-to-debug event-driven races.

2. Overview

VALERA consists of an API interception component and a runtime component. App instrumentation, achieved via bytecode rewriting, is used to intercept the communication between the app and the Android framework to produce log files. The runtime component is a manually instrumented Android framework which is used to log and replay the event schedule.

Automatic Interception through App Rewriting. VALERA leverages Redexer [3] (an off-the-shelf Dalvik bytecode rewriting tool) to instrument the app. Given the original app (APK file) along with an *Interceptor specification*, VALERA is able to find all the callsites in the bytecode that match the specification and should be intercepted. Finally, the dynamic intercepting modules and stubs are passed on to the *Redexer* to effect the interception, and repackages the bytecode into an instrumented APK.

Recording and Replaying the Event Schedule. Android is an event-driven system. Events can be classified into two groups: external and internal. External events come from the underlining hardware while internal events are posted by different threads. VALERA records the event schedule by logging each event and event processing operation into a trace file. Each event, either internal or external, is assigned a Lamport timestamp (logic order number) in the schedule. During replay, VALERA replays the events in the recorded order.

Effectiveness and Efficiency. VALERA is effective: we were able to replay 50 popular apps (most of them have in excess of 10 million installs) that use a variety of sensors. Experiments show that VALERA imposes just 1.01% time overhead for record, 1.02% time overhead for replay, 208KB/s space overhead, on average, and can sustain event rates exceeding 1,000 events/second.

3. Improving Development Lifecycle

VALERA has many applications in Android development; we present several.

3.1 Reproduce Bugs

Reproducibility is key to bug fixing. VALERA provides significant help for reproducing bugs by recording executions until a bug is encountered and then replaying the trace. We test VALERA's bug reproducing capabilities using actual bugs in popular apps. The bugs vary from different categoryies such as incorrect file format, invalid input, stress testing, and errorous scripts or plugins.

3.2 Fast Forwarding

For long-running executions, it is particularly useful to quickly reach the error state. VALERA can alter execution time without changing app behavior. This is achieved by reducing time delays between input events and gestures during data entry (e.g., virtual and physical keyboard) or idle time (e.g., user reading the screen).

3.3 Semantic Sensor Data Alteration

To test the stability of an app, VALERA also provides features to alter the recorded sensor data in a semantically meaningful way. For example, VALERA can alter GPS readings (e.g., inject a null location object to simulate GPS hardware exceptions), blur or darken the pictures captured by camera to emulate different physical environments, or change the sample rate of the audio data to test the reaction of the app towards different audio qualities.

3.4 Reproducing Event-driven Races

Event-driven races in Android are hard to debug and reproduce by current record-and-replay tools. With deterministic event order recorded, VALERA can help reproduce these races by preserving the exact event ordering. Our experiments show that VALERA can do this effectively on several open source apps: we were able to reproduce harmful event-driven races that crash the app (e.g., Tomdroid) or lead to incorrect GUI view state (e.g., NPR News).

4. Conclusions

We have presented VALERA, an approach and tool for versatile, low-overhead, record-and-replay of Android apps. Experiments with using VALERA on popular apps from Google Play, as well as replaying event race bugs, show that our approach is effective, efficient, and widely applicable to various development tasks such as testing, finding and fixing bugs.

References

[1] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. Revirt: enabling intrusion analysis through virtual-machine logging and replay. In *OSDI'02*.

[2] Y. Hu, T. Azim, and I. Neamtiu. Versatile yet lightweight record-and-replay for android. In *OOPSLA'15*.

[3] Jinseong Jeon and Kristopher Micinski and Jeffrey S. Foster. Redexer. `http://www.cs.umd.edu/projects/PL/redexer/index.html`.

[4] L. Gomez, I. Neamtiu, T.Azim, and T. Millstein. Reran: Timing- and touch-sensitive record and replay for android. In *ICSE '13*.

[5] M. Ronkko and J. Peltonen. Software industry survey, 2013. `http://www.softwareindustrysurvey.org/`.

[6] P. Maiya, A. Kanade, and R. Majumdar. Race detection for android applications. In *PLDI'14*.

[7] S. Narayanasamy, G. Pokam, and B. Calder. Bugnet: Continuously recording program execution for deterministic replay debugging. In *ISCA '05*.

[8] B. Zhou, I. Neamtiu, and R. Gupta. A cross-platform analysis of bugs and bug-fixing in open source projects: Desktop vs. android vs. ios. In *EASE '15*.