A Hybrid Mobile Farm for Testing Mobile Devices

Seyed Mohammadjavad Seyed Talebi*, Ardalan Amiri Sani*, Zhiyun Qian†

*University of California, Irvine, †University of California, Riverside
mjavad@uci.edu, ardalan@uci.edu, zhiyunq@cs.ucr.edu

Extended Abstract

Android is the most widely used operating system for mobile devices, such as smartphones, tablets, and wearables. Therefore, analyzing android and android applications for correctness, efficiency, and security is vital. Today, the Android operating system running on mobile devices is increasingly diverse. This is mainly because Android is heavily customized at multiple layers by vendors such as Samsung, HTC, SONY, and even mobile carriers (for radio layer interfaces) [2], mainly due to the difference in hardware and set of I/O devices integrated and also to provide brand specific value-added services, such as built-in application and services. This diversity poses serious challenges for testing and analysis of software running on these mobile systems.

Unfortunately, code review and static analysis is not enough for testing the software running on diverse mobile devices because these well-studied methods have important limitations, such as vulnerability to code obfuscation, not supporting analysis of both Java and native code, and vulnerability to dynamic code loading. Therefore, dynamic analysis is needed to overcome these limitations [1]. In order to perform dynamic analysis, we need a framework to run the system and control all layers of its software stack. Most existing frameworks choose to run their system's software on an emulator. Although it is straightforward to have full control on all layers of software stack running on emulators, it does not allow to find device specific bugs and vulnerabilities because the code running on an actual mobile device is different from that running in an emulator. To address this challenge, mobile farms are used to perform dynamic analysis on diverse mobile devices. Existing mobile farms run the entire software stack in mobile devices, which results in poor consolidation of resources and hence high price for users.

Our goal in this work is to make testing and analysis of the entire software stack of mobile systems running Android affordable, fast, and efficient while requiring low engineering effort, all despite the diversity of mobile software and hardware. We propose hybrid mobile farm, a cloud-based service that achieves the aforementioned goal through virtualized mobile system instances, which are virtual machines running on powerful servers that fully resemble the hardware and software configurations of various mobile systems. By running virtualized mobile instances, it is possible to increase resource consolidation and consequently reduce the cost for the user. In addition, running significant portion of software stack on virtual machines remarkably reduces the engineering effort to trace and control the running software.

Figure 1: Hybrid mobile farm architecture.

We plan to achieve our mentioned goal by conducting three fundamental contributions. (*i*) We propose to use remote I/O access to a limited set of physical mobile systems in order to enhance the fidelity of the proposed virtual mobile instances; (*ii*) we investigate virtualizing the I/O devices on mobile systems in order to share a single mobile system between many virtual instances; and (*iii*) we target at dynamic synthesis of virtual mobile instances out of the I/O devices of different physical mobile systems or standalone components to further increase the consolidation. Figure 1 shows a high-level view of our solution.

1. REFERENCES

[1] B. Reaves, J. Bowers, S. A. Gorski III, O. Anise, R. Bobhate, R. Cho, H. Das, S. Hussain, H. Karachiwala, N. Scaife, B. Wright, K. Butler, W. Enck, and P. Traynor. &Ast;Droid: Assessment and Evaluation of Android Application Analysis Tools. *ACM Comput. Surv.*, 49(3):55:1–55:30, 2016.

[2] X. Zhou, Y. Lee, N. Zhang, M. Naveed, and X. Wang. The peril of fragmentation: Security hazards in android device driver customizations. In *2014 IEEE Symposium on Security and Privacy*, pages 409–423, 2014.