Abstract

The advent of mobile devices has revolutionized many aspects of the software lifecycle. Unlike Web applications, which delegate most of the business logic to the server and use the client side for the presentation logic, mobile apps are client intensive. Another crucial difference is that the client side of Web applications is typically written using a combination of platform-independent Web languages, whereas most mobile apps have native clients written in platform-specific languages.

Though nativity hinders the portability of mobile apps across different platforms and even different devices inside the same platform, it enables smooth high-fidelity experience, high performance and compliance with the platform’s UI style—requirements that can only be satisfied natively. The main cost, beyond the initial coding of a mobile app, is to maintain its different variants in the presence of updates. Indeed, mobile code is typically updated frequently, with bug fixes and new features integrated into each new version. In the enterprise setting, these new features often revolve around security, analytics and Mobile App Management (MAM).

This paper presents Enceladus, an app-level instrumentation framework that addresses these high-maintenance costs by transparently enriching any mobile enterprise app with new analytics, security and MAM capabilities not otherwise present in the original app source code. With Enceladus, the mobile app lifecycle is significantly reduced because the instrumentation is visually configurable, and any change to the instrumentation policy can be pushed transparently without requiring a full app update.

Categories and Subject Descriptors D.2.9 [Software Engineering]: Management—Life cycle, Productivity, Software configuration management

Keywords Mobile software; Mobile application management; Mobile security; Mobile analytics; Mobile application life cycle

1. Introduction

Unlike Web applications, mobile apps are client and UI intensive. Most of the business logic resides on the client. In order to guarantee high performance, high platform fidelity, and full compliance with the platform’s UI style, most mobile apps are native. Instead of being written in a combination of Web languages, such as HTML5, JavaScript and CSS, native mobile apps are typically written in platform-dependent languages—Java for Android and Objective-C and/or Swift for iOS—which impacts portability across platforms and even across devices in the same platform, since the same app will have to be at least partially rewritten depending on whether it will run on a smartphone, tablet or smart watch.

Because of the dynamic and competitive nature of the mobile landscape, mobile apps are updated frequently, with security fixes and new features constantly integrated into new app releases. For a mobile enterprise app, most of the updates include adding analytics, security and MAM capabilities as well as changing the relevant policies. Unfortunately, such changes require modifying the source code of the app, recompiling the app, testing it, redeploying it, and reprovisioning it to the end users’ devices, with the hope that each of those end users will accept the update. This process, which has to be repeated for every platform, severely impacts the mobile app lifecycle.

In this paper, we propose Enceladus, an app-level instrumentation framework that targets the maintenance challenge by transparently enriching any mobile enterprise app with analytics, security and MAM capabilities as well as changing the relevant policies. Unfortunately, such changes require modifying the source code of the app, recompiling the app, testing it, redeploying it, and reprovisioning it to the end users’ devices, with the hope that each of those end users will accept the update. This process, which has to be repeated for every platform, severely impacts the mobile app lifecycle.

An app’s instrumentation layer can be visually configurable since Enceladus contains a module that extracts the UI views of an app, and prompts the system administrator to indicate, on each view, the individual UI components where analytics, security or MAM should be enforced and how. This allows even non-developers (such as CIO managers and system administrators) to remotely configure which portions of a given app should be tracked, secured and managed, without having to understand or even access the app’s source code—which is particularly useful for apps or libraries that were acquired from third parties—and for which the source code is not even available—or for those written by developers who have since left the enterprise.

The instrumentation layer injected into a given app performs run-time data collection (useful for analytics as well as security tracking), and can additionally respond to commands issued by a remote server in response to any given policy enforcement. Enceladus works for both Android and iOS apps.

Existing Solutions. Enceladus has goals similar to those of My-Experience [1] and LiveLab [2] in providing a comprehensive system for monitoring and logging application usage and context. Additionally, Enceladus provides a general framework for adding en-
terprise functionality to apps without requiring the hosting devices to be jailbroken, and includes a configuration process tailored towards the creation of a visual model.

2. Analytics, Security and MAM enforcement

The Enceladus architecture, visualized in Figure 1, includes three main components: instrumentation, configuration and dashboard reporting, described in Sections 2.1, 2.2 and 2.3, respectively.

2.1 Instrumentation Framework

This component allows for post-development instrumentation of Android and iOS apps. The instrumentation layer is compiled into the app’s code. As such, in alignment with the Bring Your Own Device (BYOD) paradigm, it does not affect the underlying OS.

Since it is capable of intercepting any interaction between the app and the underlying OS, the instrumentation can be used to perform advanced logging and transmission of app-usage data for analytics and security. It can also accept commands coming from a remote MAM server and modify the behavior and UI of an app, and even enhance the app with capabilities that were not originally included in its source code, as we illustrate in Section 2.3.

Enceladus supports modifying the configuration of an app’s instrumentation without asking for end users to consent, even after the app has been provisioned to end users’ devices. This is achieved by making the instrumentation configurable through a policy database. A system administrator can remotely modify the configuration of an app’s instrumentation for a given user or set of users, and push the new configuration to the relevant app instances.

Mobile platforms require the code of any app to be completely self-contained; no additional code can be dynamically loaded and executed once the app has been installed. Enceladus does not violate this requirement because the instrumentation code becomes part of the app’s bundle during recompilation, and what can possibly be modified at run time is only the instrumentation configuration, which is data rather than executable code.

We have developed several mechanisms for transparently instrumenting mobile apps. Enceladus includes what we consider the most efficient instrumentation techniques for Android and iOS.

Android, being an extension of the Java platform, inherits from Java the class-loading mechanism. Enceladus instantiates a specialized class loader and injects it into the app’s compiled code through bytecode rewriting. For each non-final operating-system method \( m \) residing in a non-final class \( A \), the Enceladus class loader creates a subclass \( B \) of \( A \) that implements \( m \), and loads \( B \) in place of \( A \). Subclass \( B \) specializes \( m \) by adding to it data-collection, security-enforcement and MAM capabilities.

On the iOS platform, Enceladus uses the standard method swizzling framework, which allows a call to an operating-system method \( m \) to be redirected to a specialized version of \( m \). The new version is executed before control is returned to \( m \).

2.2 Visual Policy Configuration

Enceladus includes a server-side module that permits even non-developers to configure the instrumentation of an app. This module extracts the UI view (or screens) of any mobile app, collects those views on a cloud-based database, and presents those views to the system administrator for configuration.

Those views are not just mere images. For each screenshot, which represent an app view, Enceladus recursively records all the embedded UI components along with their geometrical shapes and coordinates. The system administrator can then visually highlight the UI components that should be tracked, secured, or modified at run time according to the intended policy, while Enceladus transparently maps each configuration choice made by the administrator to the relevant methods in the instrumentation. Enceladus also offers a menu for configuring resources that may not be linked to a particular UI component, such as the file system, network, location, and system preferences.

We have designed and developed two mechanisms for UI capturing: static and dynamic. The static approach requires the app’s source code, analyzes it, and based on the analysis results, attempts to reconstruct as many views as possible. The dynamic approach requires an app expert to execute the app once, in profiling mode. During profiling, the views are captured along with their embedded UI components and the pertinent information.

2.3 Dashboard

At run time, the instrumentation layer gathers data that is sent to a data-collection server in raw form. The server repackages the data and sends it to various services that have subscribed for it, such as analytics engines, security managers, and MAM services. All these services are visually represented in the Enceladus dashboard, which instantaneously reports app-related activities to all the users of a given set of enterprise apps.

A system administrator or an automatic service (for example, an anomaly-detection service integrated into the dashboard) can then issue commands to the instrumentation of a particular app instance and modify its behavior or appearance. Enceladus supports numerous MAM actions, such as wiping the app data; encrypting/decrypting data; enabling/disabling access to the network, file system, and location sensor; enabling/disabling UI components, app-specific functionalities, and copy-paste operations; and adding authentication and/or authorization at specific execution points.

3. Conclusion

In this paper, we have presented Enceladus, an app-level instrumentation framework that transparently enriches any mobile enterprise app with new analytics, security and MAM capabilities not otherwise present in the original app source code. With Enceladus, the mobile app lifecycle is significantly reduced, as an instrumentation is visually configurable, and a change in the policy can be pushed transparently without requiring a full app update.

References
