Aladdin: Automating Release of Android Deep Links to In-App Content

Yun Ma[1], Xuanzhe Liu[1], Ziniu Hu[1], Dian Yang[1], Gang Huang[1], Yunxin Liu[2], Tao Xie[3]
[1]Key Lab of High-Confidence Software Technology, MoE (Peking University), Beijing, China
[2]Microsoft Research. Beijing, China [3]University of Illinois at Urbana-Champaign, Urbana, USA
{mayun, xzl, bull, yangdian, hg}@pku.edu.cn, yunxin.liu@microsoft.com, taoxie@illinois.edu

***Abstract*—Unlike the Web where each web page has a global URL to reach, a specific "content page" inside a mobile app cannot be opened unless the user explores the app with several operations from the landing page. Recently, deep links have been advocated by major companies to enable targeting and opening a specific page of an app externally with an accessible uniform resource identifier (URI). In this paper, we present an empirical study of deep links over 20,000 Android apps, and find that deep links do not get wide adoption among current Android apps, and non-trivial manual efforts are required for app developers to support deep links. To address such an issue, we propose the Aladdin approach and supporting tool to release deep links to access arbitrary locations of existing apps. We evaluate Aladdin with popular apps and demonstrate its effectiveness and performance.**

***Keywords*-Deep link; Android apps; Program analysis.**

I. Introduction

One significant feature of the Web is that there are zillions of hyperlinks to access web pages and even to a specific piece of "*deep*" web content. In the current era of mobile computing, apps have been the dominant entrance to access the Internet rather than web pages. However, mobile apps historically lack the consideration of hyperlinks. Accessing a specific in-app content requires users to launch this app and land on the "home" page, locate the page/location containing the content by a series of actions such as search-and-tap and copy-and-paste, and finally reach the target. Compared to the Web, the support such as "hyperlinks" is inherently missing so that users have to perform tedious and trivial actions. Other benefits from traditional web hyperlinks are naturally missing as well.

Realizing such a limitation, the concept of "**Deep Link**" has been proposed to enable directly opening a specific page/location inside an app from outside of this app by means of a uniform resource identifier (URI) [6]. Intuitively, deep links are "hyperlinks" for mobile apps. Currently, Google [5], Facebook [4], Baidu [2], and Microsoft [3] strongly advocate the concept of deep links, and major mobile OS platforms such as iOS [7] and Android [1] encourage their developers to release deep links. With deep links, mobile users can directly navigate into a specific page/location of apps installed on the device.

Essentially, releasing deep links for apps is to support programmable execution of apps to a specific location. However, manually implementing programmable execution to all the locations inside an app is not possible due to the high complexity of current apps. To address such a challenge, in this paper, we propose ***Aladdin***, a novel approach that can help developers automate the release of deep links for Android apps based on a cooperative framework. Our cooperative framework combines static analysis and dynamic analysis as well as engaging minimal human efforts to provide inputs to the framework for automation. Different from related efforts such as uLink [8], Aladdin requires minimal developer efforts and no intrusion to apps' original code.

II. Empirical Study of Deep Links

In this section, we present an empirical study to understand the state of the art of deep links. In current Android apps, deep links are implemented based on implicit intents, i.e., activities that support deep links **must** have a special kind of intent filters declared in the `AndroidManifest.xml` file. Such kind of intent filters should use the `android.intent.action.VIEW` with the `android.intent.category.BROWSABLE` category. We denote these intent filters as deep-link related. Therefore, we can simply take the number of activities with deep-link related intent filters as an indicator to estimate the number of deep links for an Android app.

• **Deep links are becoming popular**. We choose top 200 apps ranked by Wandoujia, a leading Android app store in China. We crawl each app's first version that could be publicly found and its latest version published on its official website as of August 2016. We compare the number of deep links of the two versions of each app. Figure 1 shows the change of the number of deep links across these two versions. Generally, it is observed that when apps are first released, only about 35% of apps have deep links. In contrast, more than 87% of these apps have supported deep links in their latest versions. Such a change indicates that the popularity of deep links keeps increasing in the past few years.

• **The coverage of deep links is still low**. We then study the latest versions of top 20,000 apps ranked by Wandoujia. About 73% of the apps do not have deep links, while 18% of the apps have only one deep link. Such result indicates that deep links are not well supported in a large scope of current Android apps. We also compute the ratio of deep-link-supported activities against the total number of activities. The median percentage is just about 5%, implying that a very small fraction of the pages can be actually accessed via deep links.

• **Supporting deep links requires high developer efforts**. We search on GitHub with the keyword "*deep link*" among all the code-merging requests in the Java language. There are totally 4,514 results returned. After manually examining the results, we find 8 projects that actually add deep links in their code-commit history. We observe that adding one deep link

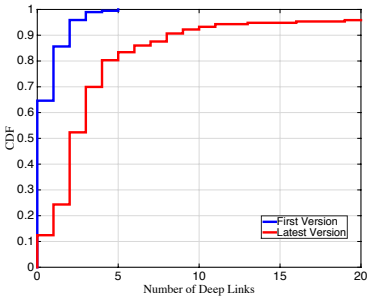

Fig. 1. Trend of apps with deep links.

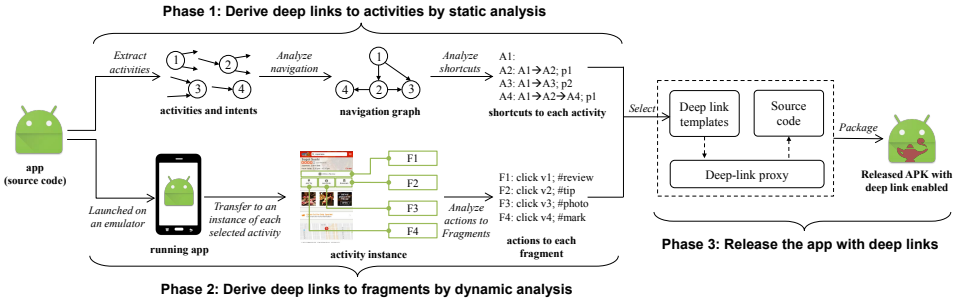

Fig. 2. Approach Overview.

requires 45∼411 LoC changes. We find that a large number of changes attribute to the refactoring of app logics to enable an activity to be directly launched without relying on other activities. Such a factor could be a potential reason why deep links are of low coverage.

III. Approach

The findings of our empirical study demonstrate the **low coverage** and **non-trivial developer efforts** of supporting deep links in current Android apps. To improve coverage, one possible solution is to leverage program analysis of apps to extract how to reach locations of in-app contents pointed by deep links. However, static analysis of Android apps can build structure relations among activities but cannot analyze dynamic fragments inside an activity; dynamic analysis of Android apps can analyze dynamic fragments but suffer from low coverage of activities, i.e., only a small fraction of activities can be reached by dynamic analysis.

To address the challenge, we propose a cooperative framework and design a tool ***Aladdin*** to automate the release of deep links. Our cooperative framework combines static analysis and dynamic analysis while minimally engaging developers to provide inputs to the framework for automation. Figure 2 shows the overview of our approach, and the workflow is illustrated as follows.

Given the source code of an app, Aladdin first derives deep-link templates that record the scripts of how to reach arbitrary locations inside the app. Each template is associated with an activity and consists of two parts: one part is the intent sequence extracted based on static analysis, recording how to reach the activity from the main activity of the app; the other part is the action sequence extracted based on dynamic analysis, recording how to reach fragments in the activity from the entrance of the activity. After developers configure to verify the activities and fragments to be deep linked, the corresponding templates and a deep-link proxy are automatically generated, and are packaged with the original source code of the app as a new `.apk` file. Each template has a URI schema used to populate a concrete deep link. The deep-link proxy provides a replay engine that can replay the sequences at runtime to execute the deep links.

When a deep link is requested with a URI conforming to a schema, the deep-link proxy is triggered to work. The corresponding template is instantiated by assigning values to parameters in the template. Then the proxy first makes the app transfer to the target activity by issuing the intents one by one in the intent sequence. Then the proxy makes the activity transfer to the target fragment by performing the actions one by one in the action sequence. Finally, the target location could be reached. Such a proxy-based architecture does not modify any original business logic of the app and is coding-free for developers.

IV. Results

We evaluate Aladdin on 20 apps chosen from Google Play. We apply Aladdin to these apps to release deep links for activities[1]. Although the current number of deep links in these apps is very low (the median is 0), Aladdin can derive deep links for all the apps (with median of 9). In particular, all the activities of 5 apps out of the total 20 apps can have deep links after being analyzed by Aladdin, indicating 100% coverage.

V. Conclusion

Although more and more apps have supported deep links, the coverage of deep links is still very low and implementing deep links requires non-trivial developer efforts. To address the issues, we propose Aladdin, a novel approach to automatically release deep links for Android apps. The evaluations on 20 apps demonstrate that the coverage of deep links can be increased by 52% on average.

Acknowledgments

This work was supported by the National Basic Research Program (973) of China under Grant No. 2014CB347701, the Natural Science Foundation of China (Grant No. 61370020, 61421091, 61528201, 61529201), the Microsoft-PKU Joint Research Program, and NSF under grants no. CCF-1409423, CNS-1434582, CNS-1513939, CNS-1564274.

References

[1] App links in android 6. https://developer.android.com/training/app-links/index.html.
[2] Baidu app link. http://applink.baidu.com.
[3] Bing app linking. https://msdn.microsoft.com/en-us/library/dn614167.
[4] Facebook app links. https://developers.facebook.com/docs/applinks.
[5] Google app indexing. https://developers.google.com/app-indexing/.
[6] Mobile deep linking. https://en.wikipedia.org/wiki/Mobile_deep_linking.
[7] Universal links in iOS 9. https://developer.apple.com/library/ios/documentation/General/Conceptual/AppSearch/UniversalLinks.html.
[8] T. Azim, O. Riva, and S. Nath. uLink: Enabling user-defined deep linking to app content. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2016*, pages 305–318, 2016.

[1]Please refer to http://sei.pku.edu.cn/~mayun11/aladdin/ for more details.