2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

Understanding Third-party Libraries in Mobile App Analysis

Haoyu Wang
School of Computer Science,
Beijing University of Posts and Telecommunications,
Beijing, China
Email: haoyuwang@bupt.edu.cn

Yao Guo
Key Lab of High-Confidence Software Technologies (MOE),
School of EECS, Peking University,
Beijing, China
Email: yaoguo@pku.edu.cn

***Abstract*—Third-party libraries are widely used in mobile apps. Recent studies showed that third-party libraries account for more than 60% of the code in Android apps on average. As a result, program analysis on Android apps typically requires detecting or removing third-party libraries first, because they usually introduce significant noises and affect the analysis results. In this technical briefing, we will introduce the latest research advances related to third-party libraries used in mobile apps. The briefing will be focused on: (1) the importance of third-party libraries, including the current status, types and distribution, based on the analysis results on over 1 million Android apps; (2) how to detect third-party libraries from Android apps, including an overview of existing approaches and their limitations; (3) the implications of third-party libraries in software engineering tasks such as mobile app analysis, as well as case studies from the domain of program analysis and mobile security; (4) future challenges and research directions related to third-party libraries.**

***Keywords*-Third-party libraries; mobile apps; program analysis; mobile security**

I. Introduction

Third-party libraries (TPLs) are widely used in mobile apps. For example, app developers often use advertising libraries as a source of revenue, or integrate social networking libraries to simplify the login process. Previous research showed that ad libraries are used in almost two-thirds of the popular apps from Google Play [14]. Other libraries such as social network libraries and mobile analytic tools are also very popular. Some apps even use more than 20 TPLs [1]. TPLs account for more than 60% of the code in Android apps on average [15].

Because TPLs account for a large portion of the code in Android apps, mobile app analysis typically should detect or remove TPLs first or analyze TPLs separately. For a variety of app analysis tasks, including permission gap analysis, mobile app clone detection and app behavior analysis, TPLs usually introduce significant noise and they could greatly affect the analysis results. For example, TPLs account for a large portion of permission uses in mobile apps[1], which is one of the main reasons that previous work found the huge gap between app description and requested permissions [12]. In app clone detection and app behavior analysis, most approaches need to extract code-level features (e.g., API invocations) from decompiled apps. The analysis results will differ significantly if TPLs are not correctly removed.

Furthermore, TPLs bring in new security and privacy risks. Even the most popular libraries may pose threats to the privacy of mobile users. For example, some libraries exploit the privileges of their host apps, track users in order to provide efficient targeted ads, and collect user data to extort benefits. Many popular libraries have been found severe vulnerabilities [3]. For a variety of research on mobile app security analysis, it is important to separate TPLs from app custom code and determine whether the sensitive behaviors are introduced by libraries or app custom code.

As a result, many studies have focused on how to identify TPLs from mobile apps, as well as how to use the TPL analysis results in tasks such as mobile app analysis, security and privacy analysis, or app protection.

II. Relevance to the SE community

A large amount of research efforts have been focused on mobile app analysis, and *mobile app* is also one of the most accepted topics according to the statistics of ICSE 2016. Many research tasks have considered the importance of TPLs.

- ***App Clone and Similar App Detection.*** TPL is a key factor that impacts detection accuracy, thus almost all approaches have attempted to eliminate the impact of libraries. For example, Centroid [4] uses a white-list based approach to filter TPLs, WuKong [15] uses a clustering-based approach to filter TPLs. The result of CLANDroid [8] suggested significant accuracy difference when TPLs are excluded.
- ***Abnormal Behavior Analysis.*** CHABADA [5] detects inconsistencies between app description and app functionalities. Their results suggested that TPLs are one of the key factors that lead to outlier apps. WHYPer [12] analyzes the inconsistencies between app description and requested permissions. TPLs should be one of the main reasons that lead to the huge gap between app description and requested permissions, because TPLs account for a large portion of permission uses in mobile apps [1].
- ***Empirical Studies.*** Hassan *et al.* [13], [11] have conducted empirical studies on TPLs, including analyzing the relationship between TPLs and app quality, as well

[1]For apps that use ACCESS_FINE_LOCATION, more than 60% of them use it in TPLs [1].

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.161

as framework reuse analysis in Android apps. Other work [7] also demonstrated the importance of TPLs in code reuse analysis of mobile apps.

- ***Security and Privacy Analysis.*** It is important to separate TPLs from app custom code and analyze whether the sensitive behaviors are introduced by libraries or custom code. LIBSCOUR [3] quantified the security impact of TPLs on the Android ecosystem. DroidAPIMiner [2] demonstrated that TPLs could greatly affect the detection results of API feature based malware detection. Lin *et al.* [6] proposed to infer the purpose of permission use by analyzing sensitive data used in TPLs. Another line of research is focused on the privilege separation for mobile apps and TPLs [9].

III. Technical Briefing Content

This technical briefing aims to introduce researchers and app developers to the latest research advances in TPL analysis, as well as how these new approaches can be used to support a variety of mobile app analysis tasks. The technical briefing will be composed of four parts:

1) We will first highlight the importance of TPLs, including the current status, different types and distributions of TPLs. We will also dissect the usage of TPLs of over 1 million Android apps from Google Play, including detailed analysis on the most popular libraries.
2) We will give an overview of the current approaches on TPL detection, including approaches based on *whitelists*, *machine learning*, *semantic block clustering*, *package level clustering*, *SDK features* and *rule-based* approaches. We will discuss the pros and cons for each approach from the aspects of *library coverage*, *detection accuracy*, *detection efficiency*, and *obfuscation-resilience*, as well as usage scenarios and limitations.
3) We then discuss how TPL analysis could be used to improve a variety of mobile app analysis tasks, ranging from permission gap analysis of mobile apps, static analysis, repackaged app detection, malware detection, and fine-grained access control. We will emphasize the importance of TPL analysis on several case studies from the domain of program analysis and mobile security.
4) Finally, we will highlight future challenges and research directions related to TPLs.

IV. Target Audience

This technical briefing is suitable for both researchers, students and app developers in general. For researchers, an updated state of the art will be exposed, the importance of TPLs on app analysis will be presented based on scientific grounds, and various tools will be discussed to help them identify and filter TPLs. For app developers, a complete and detailed analysis on TPLs will be presented, which could help them make informed decisions on how to select TPLs during app development. No prior knowledge of TPLs is required. The audience will be provided a list of further readings and available tools.

V. Author Biographies

Haoyu Wang is currently an assistant professor in School of Computer Science, Beijing University of Posts and Telecommunications. He received his PhD degree in Computer Science from Peking University in 2016. He was a visiting PhD student at Carnegie Mellon University for one year. His research interest lies at the intersection of program analysis, privacy and security, and mobile systems. In his recent work [10], [15], he has analyzed over 1 million Android apps, including how to detect, classify and use third-party libraries.

Yao Guo is currently an associate professor in Department of Computer Science, Peking University. He received his PhD in Computer Engineering from University of Massachusetts at Amherst in 2007. His research interests include mobile systems and applications, low-power design and software engineering. He has worked on related topics such as aspect recommendation, clone detection and mobile app analysis. He has led the effort in LibRadar [10], which has become a widely used open-source tool for detecting third-party libraries.

Acknowledgment

This work is partly supported by the Funds for Creative Research Groups of China (No. 61421061), the National Natural Science Foundation of China (No. 61421091), and the Beijing Training Project for the Leading Talents in S&T (No. ljrc 201502).

References

[1] Privacygrade: Grading the privacy of smartphone apps. http://privacygrade.org/.
[2] Y. Aafer, W. Du, and H. Yin. Droidapiminer: Mining api-level features for robust malware detection in Android. In *SecureComm'13*, pages 86–103.
[3] M. Backes, S. Bugiel, and E. Derr. Reliable third-party library detection in Android and its security applications. In *CCS'16*.
[4] K. Chen, P. Liu, and Y. Zhang. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets. In *ICSE'14*.
[5] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In *ICSE'14*.
[6] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *UbiComp'12*, pages 501–510.
[7] M. Linares-Vásquez, A. Holtzhauer, C. Bernal-Cárdenas, and D. Poshyvanyk. Revisiting android reuse studies in the context of code obfuscation and library usages. In *MSR'14*, pages 242–251.
[8] M. Linares-Vasquez, A. Holtzhauer, and D. Poshyvanyk. On automatically detecting similar android apps. In *ICPC'16*.
[9] B. Liu, B. Liu, H. Jin, and R. View. Efficient privilege de-escalation for ad libraries in mobile apps. In *MobiSys'15*.
[10] Z. Ma, H. Wang, Y. Guo, and X. Chen. Libradar: Fast and accurate detection of third-party libraries in android apps. In *ICSE'16*.
[11] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. E. Hassan. A large-scale empirical study on software reuse in mobile apps. *IEEE Software*, 31(2):78–86, 2014.
[12] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie. Whyper: Towards automating risk assessment of mobile applications. In *SEC'13*, pages 527–542.
[13] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan. Impact of ad libraries on ratings of android mobile apps. *IEEE Software*, 31(6):86–92, 2014.
[14] N. Viennot, E. Garcia, and J. Nieh. A measurement study of Google Play. In *SIGMETRICS'14*, pages 221–233.
[15] H. Wang, Z. Ma, Y. Guo, and X. Chen. Wukong: A scalable and accurate two-phase approach to Android app clone detection. In *ISSTA'15*, 2015.