M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 117

CodeMatch: Obfuscation Won't Conceal Your Repackaged App

Leonid Glanz, Sven Amann, Michael Eichberg, Michael Reif, and Mira Mezini[1]

Abstract: Popular mobile apps are regularly installed by millions of users. This fact attracts malicious actors to create altered, repackaged versions of those apps to steal the original owner's revenue or to trick users to infect their devices with malware. Detecting such repackaged apps is, therefore, necessary for a secure and viable app market but is challenging due to the use of code obfuscation and the widespread usage of libraries. Due to the recent fact, non-repackaged, legitimate apps often share a majority of their code base and are classified as repackaged by state-of-the-art detectors. We, therefore, propose a new library filtering approach that relies on code representations at five different abstraction levels to achieve resilience against code obfuscation. Additionally, we propose to use the most abstract representation in combination with fuzzy-hashing to detect repackaged apps. Our evaluation shows that the overall approach leads to a better detection rate up to 50%.

Keywords: library detection; repackage detection; obfuscation; code analysis

1 Overview

Since 2012 several techniques for repackage detection have been proposed and can be broadly classified as being code-agnostic, graph-based, user-interface-based, and code-signature-based. A challenge for all these repackage detectors is code transformation. Developers regularly minify and optimize their apps to increase performance. Additionally, they obfuscate their apps to protect their intellectual property. However, attackers also apply obfuscation to hide malicious code and to evade signature-based detectors, such as anti-virus software.

Current repackage detectors can only handle weak forms of obfuscation such as one-by-one renaming, but more sophisticated ones, are not supported, e.g., which change the defining package of classes. However, at least 20% [Gla17] of the apps found in Google Play Store [Ink17] use such techniques. The prevalent reuse of libraries in apps further inhibits the effectiveness of current detectors. Wang et al. [WGMC15] reported that more than 60% of the sub-packages in apps belong to libraries. Hence, separating the libraries from the app code is necessary. Otherwise, detectors wrongly flag apps as repackages which use similar libraries because they share a large code base. Another challenge for repackage detectors are apps generated by App Makers, e.g., apps-builder [Bui17]. In that case, the vast majority

[1] Technische Universität Darmstadt, Software Technology Group, Hochschulstraße 10, 64289 Darmstadt, Deutschland {glanz|amann|eichberg|reif|mezini}@cs.tu-darmstadt.de

of the code base will be the same, and the rest will still be very similar. Current approaches will falsely flag such apps as repackaged.
To address the challenges, we propose a library filter LibDetect and an app matcher CodeMatch, whereby the latter uses the former. LibDetect uses five hierarchically organized representations which enable an adequate precision/recall trade-off. If a library method is only weakly obfuscated, LibDetect will identify the method using a less abstract representation when compared to stronger obfuscated methods. After library filtering, our app matcher CodeMatch uses our most resilient representation as a foundation, sorts the output and performs fuzzy hashing [Kor06] on top of it to withstand various sophisticated obfuscation techniques.
For the evaluation, we built ground truths for library filtering and repackage detection by manually inspecting thousands of apps. Afterwards, we executed two library filters, and four repackage detectors to compare them with our approaches. Additionally, we executed CodeMatch with different library filters to evaluate its independent detection quality.

2 Conclusion

We presented an approach to detect repackaged apps, which relies on abstract code representations which are obfuscation-resilient. The approach consists of two steps (1) a new advanced library detection approach and (2) the fuzzy hashing of the app's code. Our evaluation shows that we can identify up to 50% more repackaged apps than the state-of-the-art. These results are due to LibDetect which correctly filters up to 70% more libraries than previous approaches.
Our implementation and all evaluation data is available at [Gla17].

References

[Bui17] A. Builder. *Apps Builder*. `http://www.apps-builder.com`. Lastaccessed: 01/11/2017. 2017 (cit. on p. 11).

[Gla17] L. Glanz. *CodeMatch Artifacts*. `http://www.st.informatik.tu-darmstadt.de/artifacts/codematch/`. Last Accessed: 06/30/2017. 2017 (cit. on pp. 11, 12).

[Ink17] G. Ink. *Google Play Store*. `https://play.google.com/store?hl=us`. Last Accessed: 12/04/2017. 2017 (cit. on p. 11).

[Kor06] J. Kornblum. "Identifying almost identical files using context triggered piecewise hashing". In: *Digital investigation* 3 (2006), pp. 91–97 (cit. on p. 12).

[WGMC15] H. Wang, Y. Guo, Z. Ma, and X. Chen. "Wukong: A scalable and accurate two-phase approach to android app clone detection". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM. 2015, pp. 71–82 (cit. on p. 11).