Transforming and Taming Privacy-Breaching Android Applications

Mu Zhang Heng Yin
Syracuse University
{*muzhang, heyin*}*@syr.edu*

Privacy concerns in the Android platform are increasing. Previous studies [1]–[5] have exposed that both benign and malicious Apps are stealthily leaking users' private information to remote servers. By design, Android security model is based on permissions. At install time, an App requests a set of permissions. In order to use this App, the user has to grant all the requested permissions, or refuse to install it. Once granted, permissions are not revocable. The Android system can thus no longer prevent an App from misusing permissions and violating the user's privacy.

Prior research efforts have been made to ameliorate the privacy leakage problem. Some earlier solutions extended Android's install-time constraints and enriched Android permissions [6], [7]. Some aimed at enforcing permissions in a finer-grained manner and in a more flexible way [8]–[11]. In addition, some efforts were also made to improve isolation on various levels, so that each isolated component could be assigned with a different set of permissions [12]–[14].

In principle, all these solutions enforce various policies at individual checkpoints. These approaches deploy checkpoints at certain execution points. Fundamentally, privacy leakage is an information flow security problem. To directly address this problem, Hornyack et al. proposed AppFence to enforce information flow policies at runtime [3]. With support of TaintDroid [2], AppFence keeps track of the propagation of private information. Once privacy leakage is detected, AppFence either blocks the leakage at the sink or shuffle the information from the source.

Though effective in terms of blocking privacy leakage, AppFence has several limitations: 1) Due to the taint tracking on every single Dalvik bytecode execution, AppFence incurs significant performance overhead; 2) As firmware modification is required, deploying it on large amount of Android devices can be challenging.

To address these limitations, we propose a bytecode transformation approach to effectively defeating privacy leakage in Android applications. Given an unknown Android App, we first perform application-wide static dataflow analysis to identify potential taint propagation chops for information leakage. Then to keep track of taint propagation and prevent the actual information leakage, we insert bytecode instructions along the taint propagation chops. To further improve performance, we apply a series of optimization methods to remove redundant and unnecessary instrumentation bytecode. As a result, we transform an App with potential privacy leakage into a more secured App, in which privacy leakage is disabled. Compared to dynamic taint analysis approach like AppFence, our bytecode transformation approach has the following advantages: 1) the deployment of our approach is simple, as we only transform the original App to enforce certain information flow policies and no firmware modification is needed; 2) the performance overhead of our approach is minimal, due to the static dataflow analysis in advance and numerous optimizations that are applied to the instrumentation code.

We implement a prototype in 16 thousand lines of Java code, based on the Java bytecode optimization framework Soot [15]. We leverage Soot's capability to perform static dataflow analysis and bytecode instrumentation. We evaluate our tool on 16 real-world privacy-breaching Android Apps. Our experiments show that transformed program runs correctly after instrumentation, while privacy leakage is effectively eliminated.

References

[1] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proceedings of the 20th Usenix Security Symposium*, August 2011.

[2] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of OSDI*, 2010.

[3] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications," in *Proceedings of CCS*, 2011.

[4] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, May 2012.

[5] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets," in *Proceedings of NDSS*, February 2012.

[6] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically rich application-centric security in android," in *Proceedings of ACSAC*, 2009.

[7] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of CCS*, 2009.

[8] M. Conti, V. T. N. Nguyen, and B. Crispo, "Crepe: context-related policy enforcement for android," in *Proceedings of the 13th International Conference on Information Security*, 2011.

[9] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in *Proceedings of the 4th international conference on Trust and trustworthy computing*, 2011.

[10] M. Nauman, S. Khan, and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 2010.

[11] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: trading privacy for application functionality on smartphones," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, 2011.

[12] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4android: a generic operating system framework for secure smartphones," in *Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices*, 2011.

[13] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, "Cells: a virtual mobile smartphone architecture," in *Proceedings of SOSP*, 2011.

[14] S. Shekhar, M. Dietz, and D. S. Wallach, "Adsplit: Separating smartphone advertising from applications," in *Proceedings of the 20th Usenix Security Symposium*, August 2012.

[15] Soot: a java optimization framework. [Online]. Available: http://www.sable.mcgill.ca/soot/