Easily Instrumenting Android Applications for Security Purposes

Eric Bodden Secure Software Engineering Group
European Center for Security and Privacy by Design (EC SPRIDE)
Technische Universität Darmstadt & Fraunhofer SIT
Darmstadt, Germany
eric.bodden@ec-spride.de

ABSTRACT

With the increasing market share of the Android operating system, the platform also becomes more and more interesting for adversaries. Current malware on the Android platform exhibits a very diverse spectrum of malicious behaviors such as sending text messages to costly premium rate numbers [6,21], infecting connected computes with malware [12], or abusing the phone's resources for bot networks [16,20,21]. In addition, many smartphones users store highly privacy-sensitive data like calendars, personal photos, SMS messages, e-mails and more on their phones. Adversaries can attempt to access this data without the user's consent for the purpose of (industrial) espionage or targeted advertisement campaigns.

Such attacks are usually carried out using malicious applications which, e.g., pretend to be a game, but also contain malicious code embedded by the original developer. Such attacks are successful since users can hardly estimate the trustworthiness of applications offered in the different markets. Various approaches for analyzing the behavior of untrusted applications have thus been proposed, both static and dynamic [4, 5, 7–11, 13, 15, 18, 19]. While static approaches attempt to assess a program in a black-box fashion by looking at its (binary) code, dynamic analysis monitor the actual execution of the target program.

In this tutorial, we will focus on dynamic instrumentation, i.e., program rewriting techniques. Instrumenting the program gives access to full runtime information, e.g., user inputs and environment settings, which greatly increases the precision of detected policy violations. Additionally, countermeasures can directly be taken by, e.g., blocking the call to a costly premium rate number of asking the user for consent before proceeding.

We will show three different possibilities for instrumenting Android applications. Firstly, we will use the AspectBench compiler (abc) [1] to declaratively define instrumentations using AspectJ. We will then extend this approach to Tracematches [2] allowing finite-state policies to be expressed (e.g., no more than three SMS messages to the same phone number, to prevent spam). We will show where such techniques are applicable based on practical examples from the Android world and will also point out where their respective limits are.

Finally, we will show how to manually instrument applications by loading them into Soot [14], directly manipulating the *Jimple* intermediate representation and then producing a new application package containing the modifications for maximum flexibility both in terms of program changes and the policies to be enforced. We will use this technique to solve example problems that were not applicable to AspectJ and Tracematches.

Further background information on abc and Soot as well as general remarks on platform-specific aspects of program instrumentation on Android will also be given. Unlike system-level approaches like TaintDroid [5], instrumenting the application's bytecode at the application level requires no modifications to the operating system and does not require root access to the device either. The instrumented applications can be installed and used as usual with no visible difference to the user, only differing in behavior when an instrumented action is triggered.

CCS'13, November 4–8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11.
http://dx.doi.org/10.1145/2508859.2516759.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls

General Terms

Security

Keywords

Android, Runtime Enforcement, Instrumentation, Dynamic Analysis

1. AUDIENCE AND PREREQUISITES

This tutorial is intended for security researchers interested in instrumenting or rewriting applications on the Android platform. Previous knowledge of Android application development is helpful but not a strict requirement, as example applications and a short introduction will be given. For the parts on AspectJ and Tracematches, basic knowledge of the respective concepts is advantageous. However, the examples given in the tutorial can also be used as a starting point for approaching the topic.

2. PROPOSED TIMELINE

We propose the following rough timeline:

- Background: The Android platform and its application architecture
- Background: The Android SDK and its tools for developers
- Background: Soot and its Jimple intermediate representation
- Background: Conversion of Android's dalvik bytecode into Jimple and back
- Hands-on: Converting an app into Jimple and inspecting its code
- High-level instrumentation: Using tracematches to instrument Android code
- Hands-on: Tracematch instrumentation, re-packaging and execution in emulator
- Limitations of tracematches
- More fine-grained instrumentation with AspectJ aspects
- Hands-on: AspectJ instrumentation, re-packaging and execution in emulator
- Limitations of AspectJ
- More fine-grained, manual instrumentation with Soot and Jimple
- Hands-on: Jimple instrumentation, re-packaging and execution in emulator
- Outlook: Combination with static analysis
- Wrap-up

3. PREVIOUS TUTORIALS

An abc tutorial was held at AOSD 2006, however, did not focus on runtime monitoring. The last tutorial on Soot was at PLDI 2003 and focused on static analysis rather than instrumentation. There has never been a tutorial on the combination of Soot, abc and Android.

Furthermore, we have been invited to give a tutorial on Java and Android instrumentation using Soot and abc at RV 2013. The CCS tutorial will be an improved iteration, re-targeted specifically for the security community.

4. BIOGRAPHY OF PRESENTERS

In the Android space, the Secure Software Engineering Group is known for its work on the FlowDroid static taint-analysis system for Android [8], the DroidBench benchmark suite [17] for Android taint analysis and the SuSi [3] tool for the semi-automatic detection of sources and sinks of private information in Android applications.

Prof. Dr. Eric Bodden. Eric is heading the Secure Software Engineering group at the Fraunhofer Institute for Secure Information Technology (SIT), the Technische Universität Darmstadt and the European Center for Security and Privacy by Design (EC SPRIDE). He is also a principal investigator at the Center for Advanced Security Research Darmstadt (CASED). Eric is heading the Emmy-Noether research group RUNSECURE, which focuses on securing dynamic program executions through dynamic instrumentation techniques. He is world renowned as the lead maintainer of Soot and has been a top contributor to the AspectBench Compiler (abc) for more than six years. More than a dozen of his most cited publications are directly based on Soot and abc. In 2009 Eric finished his doctoral dissertation at McGill University, under the supervision of Laurie Hendren, an ACM Fellow and lead scientist in the field of compiler engineering and program analysis. Eric's thesis topic was the static ahead-of-time evaluation of inlined reference monitors using abc. In the past, Eric has won two ACM Distinguished Paper Awards, both on dynamic program analysis, one of them related to data-race detection using AspectJ. In 2005 he won the worldwide ACM Student Research Competition with the topic of his Diploma thesis, which was the first work to link AspectJ to runtime monitoring.

5. REFERENCES

[1] Chris Allan, Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. abc: the aspectbench compiler for aspectj. In *Proceedings of the 4th international conference on Generative Programming and Component Engineering*, GPCE'05, pages 10–16, Berlin, Heidelberg, 2005. Springer-Verlag.

[2] Chris Allan, Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. Adding trace matching with free variables to aspectj. In *Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '05, pages 345–364, New York, NY, USA, 2005. ACM.

[3] Steven Arzt, Siegfried Rasthofer, and Eric Bodden. Susi: A tool for the fully automated classification and categorization of android sources and sinks, Mai 2013.

[4] Michael Backes, Sebastian Gerling, Christian Hammer, Matteo Maffei, and Philipp von Styp-Rekowsky. Appguard: enforcing user requirements on android apps. In *Proceedings of the 19th international conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'13, pages 543–548, Berlin, Heidelberg, 2013. Springer-Verlag.

[5] William Enck, Peter Gilbert, Byung gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, pages 393–407, 2010.

[6] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX*

conference on Security, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[7] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[8] Christian Fritz, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves le Traon, Damien Octeau, and Patrick McDaniel. Highly precise taint analysis for android applications, Mai 2013.

[9] Adam P Fuchs, Avik Chaudhuri, and Jeffrey S Foster. Scandroid: Automated security certification of android applications. *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/˜ avik/projects/scandroidascaa*, 2009.

[10] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale. In *Proceedings of the 5th international conference on Trust and Trustworthy Computing*, TRUST'12, pages 291–307, Berlin, Heidelberg, 2012. Springer-Verlag.

[11] Johannes Hoffmann, Martin Ussath, Thorsten Holz, and Michael Spreitzenbarth. Slicing droids: program slicing for smali code. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1844–1851. ACM, 2013.

[12] Victor Chebyshev (Kaspersky Labs). Mobile attacks!, 2013.

[13] Jinyung Kim, Yongho Yoon, Kwangkeun Yi, and Junbum Shin. ScanDal: Static analyzer for detecting privacy leaks in android applications. In Hao Chen, Larry Koved, and Dan S. Wallach, editors, *MoST 2012: Mobile Security Technologies 2012*, Los Alamitos, CA, USA, May 2012. IEEE.

[14] Patrick Lam, Eric Bodden, Ondrej Lhotak, and Laurie Hendren. The soot framework for java program analysis: a retrospective. In *Cetus Users and Compiler Infastructure Workshop (CETUS 2011)*, Oktober 2011.

[15] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. Chex: statically vetting android apps for component hijacking vulnerabilities. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 229–240, New York, NY, USA, 2012. ACM.

[16] Heloise Pieterse and Martin S. Olivier. Android botnets on the rise: Trends and characteristics. In Hein S. Venter, Marianne Loock, and Marijke Coetzee, editors, *ISSA*, pages 1–5. IEEE, 2012.

[17] Christian Fritz Steven Arzt and Siegfried Rasthofer. Droidbench - benchmarks, 2013.

[18] Rubin Xu, Hassen Saïdi, and Ross Anderson. Aurasium: practical policy enforcement for android applications. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 27–27, Berkeley, CA, USA, 2012. USENIX Association.

[19] Zhemin Yang and Min Yang. Leakminer: Detect information leakage on android with static taint analysis. In *Third World Congress on Software Engineering (WCSE 2012)*, pages 101–104, 2012.

[20] Yuanyuan Zeng, Kang G. Shin, and Xin Hu. Design of sms commanded-and-controlled and p2p-structured mobile botnets. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 137–148, New York, NY, USA, 2012. ACM.

[21] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 95–109, Washington, DC, USA, 2012. IEEE Computer Society.