Android Malware Analysis: from technical difficulties to scientific challenges

Jean-François Lalande

▶ **To cite this version:**

Jean-François Lalande. Android Malware Analysis: from technical difficulties to scientific challenges. SecITC 2018 - International Conference on Innovative Security Solutions for Information Technology and Communications, Nov 2018, Bucharest, Romania. pp.1-54, 10.1007/978-3-030-12942-2_2 . hal-01906318

HAL Id: hal-01906318

https://centralesupelec.hal.science/hal-01906318

Submitted on 8 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Android Malware Analysis: from technical difficulties to scientific challenges*

Jean-François Lalande

CentraleSupélec, Inria, Univ Rennes, CNRS, IRISA
F-35065, Rennes, France
jean-francois.lalande@inria.fr

Abstract. Ten years ago, Google released the first version of its new operating system: Android. With an open market for third party applications, attackers started to develop malicious applications. Researchers started new works too. Inspired by previous techniques for Windows or GNU/Linux malware, a lot of papers introduced new ways of detecting, classifying, defeating Android malware. In this paper, we propose to explore the technical difficulties of experimenting with Android malware. These difficulties are encountered by researchers, each time they want to publish a solid experiment validating their approach. How to choose malware samples? How to process a large amount of malware? What happens if the experiment needs to execute dynamically a sample? The end of the paper presents the upcoming scientific challenges of the community interested in malware analysis.

Keywords: malware analysis · mobile phones.

1 Introduction

Research about Android malware mainly addresses three types of problem: deciding if an application is a malware or not (detection), classifying a malware sample among families (classification) and explaining precisely what a sample is doing (characterization). These three types of contribution require a dataset of malware for performing experiments that support the proposed contribution. Nevertheless, performing reliable experiments is not as simple as expected. In addition with the difficulties to reproduce previous experiments, many technical difficulties can have an impact on the results.

This paper summarizes the encountered difficulties when experimenting with Android malware. In particular, we focus on the choice of the malware dataset, that can have a large influence on the outperformed results. Finally, we discuss the upcoming challenges when contributing to the field of malware analysis.

* This work has received a French government support granted to the COMIN Labs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01.

2 Designing malware analysis experiments

2.1 Datasets

The choice of the dataset is of primary importance to discuss with accuracy the obtained results. As Android is quickly evolving, malware samples are quickly outdated and some of them cannot be installed on recent versions of the operating system. Thus, fundamental papers that have a lot of citations like CopperDroid [13], IntelliDroid [15], TaintDroid [7], to name a few, suffer from outdated datasets or incompatibility problems. For example CopperDroid runs on several Android versions, which is fine, but uses three outdated datasets: Contagio mobile[1], the Genome Project [17] and samples provided by McAfee. Contagio mobile is a blog which contains few samples, posted by contributors. The Genome Project has now closed its service and the database cannot be accessed anymore. Finally, the samples obtained from a collaboration with McAfee is of course interesting but cannot be reproduced in other papers.

The size of the dataset makes experiments difficult to compare. For example, a recent paper about malicious libraries [5] provides results based on 1.3 millions of applications, whereas the experiments of TriggerScope for dealing with logic bombs evaluate the approach on 14 malware among 9,313 applications. Experiments are difficult to compare when there is so much difference between the processed number of applications. Additionally, most of the time, authors have to reimplement previous contributions, such as TriggerScope that reimplemented Kirin [8], DroidAPIMiner [1], and FlowDroid [4], which is obviously a lot of efforts.

New datasets recenlty appeared that try to address these problems. AndroZoo [3] is a large dataset of million of applications that is helpful for a having a large diversity of benign and malware applications. In particular, they provide pairs of applications that show that a lot malware applications are repackaged version of benign applications. The AMD dataset [14] is another contributions that provides 24,650 malware samples. Some contextual information (classes, actions,...) are provided that have been computed for a part by automatic tools and for the other part by a manual analysis. We believe that these two contributions for building reliable and documented datasets are an important step towards reproducible security experiments, but some questions remain, for example the lifespan of a dataset, used for training purpose [12].

2.2 Designing an experiment

From our point of view, designing an experiment for malware analysis always follows the same steps. After selecting a sample to be analyzed from the dataset, authors process this sample by applying static or dynamic analysis techniques. Depending on there goals (detection, classification, characterization), they compare the obtained performances with the results obtained by other tools. Even if this process is simple to understand, several question arise:

[1] `http://contagiominidump.blogspot.com`

Is the sample really a malware? Picking up some sample from large unlabeled datasets or the play store gives a large variety of applications, representative of the reality. If this effort is continuous with time, authors are sure to capture new malware samples that appear on the play store before being removed. Then, samples should be analyzed to decide if they are malware or not, especially when evaluating a detection algorithm. For solving this issue, most of authors rely on VirusTotal to decide if a sample is malicious or not. We believe that it is not a good idea, because the analysis tools used by VirusTotal has no reason to decide accurately [10], especially for recent samples.

Where is the payload? The answer should be provided by the dataset. Most of papers use heuristics for answering this question, for example the approach of Aafer et al. [1] that study the API used by each part of the code. The idea behind this heuristic is that the malicious parts of the code use specific APIs and can be exposed. Identifying the payload is especially important when studying multiple repackaged applications where the payload is hidden in the code of benign applications. Nevertheless, five years after Aafer et al.'s publication, the mentioned evasion techniques, reflection, native code, encryption, dynamic loading are widely used by malware. Even benign applications now use packers that use these techniques to protect their code [6].

Is the static and dynamic analysis possible? Most approaches analyze statically the sample before eventually executing it in a sandbox or a real smartphone. Few papers mention that a static and dynamic analysis can crash. As an example, Yang et al. [16] report in their paper that the executed sample crashes 76% of the time. The reasons behind these crashes remain unknown. It could be the setup of the experiment that is not reliable with the studied sample, for example a bad version of the operating systems, missing libraries such as the Google services, etc. For defeating the static analysis, malware can introduce some artifact for disturbing disassembly tools such as apktool [9].

Consequently, in the last years, we proposed several tools for automatizing the static and dynamic analysis of Android malware [11,2]. Our intention is to perform the classical steps of an analyzer: CFG extraction from a static analysis, payload identification using an heuristic similar to DroidAPIMiner, computation of execution paths, including the path that can use callbacks of the framework, and finally dynamic execution with an automatic exploration of the graphical interface. Additionally, we provide fallback methodologies when a difficulty is encountered. For example, if the Manifest file cannot be decoded, we provide a generated minimalistic one for our tools being able to run. Another example is the ability of GroddDroid to change a branching condition to force the execution flow towards a suspicious part of the code not analyzed yet.

Nevertheless, a lot of problems still exist when processing large amount of malware. The next section gives an overview of the remaining challenges.

3 Next upcoming challenges

As explained before, building reliable datasets remain a difficult challenge. Additionally, in some countries, sharing malicious codes is forbidden by the law. Some samples can also contain piggybacked code from copyrighted benign applications. These difficulties should be addressed by international consortiums that would provide a safe infrastructure for sharing resources. Such an effort could also help on the dataset labeling i.e. by determining the payload location, malware's actions, etc.

Experiments are also limited by the devices themselves. As it is hazardous to run a malware in an emulator, with anti analysis techniques, most of authors run their experiment on a real smartphone with an eventually modified operating system. In such conditions, running a clean execution of a malware, including a smartphone flashing step, requires several minutes. If several environments or parameters have to be tested the total running time becomes prohibitive.

Finally, these ten years of research efforts are intimately linked with Google's roadmap. In the near future, introducing a new operating system like Fuschia can lead to the restart of many results, at the price of few novelty. In the meantime, the ecosystems that Google try to promote has received few attention from the research community. For example, Android automotive, Wear OS has not being extensively studied and malware may exist on these platforms - if they don't already exist.

References

1. Aafer, Y., Du, W., Yin, H.: DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. Security and Privacy in Communication Networks **127**, 86–103 (2013). https://doi.org/10.1007/978-3-319-04283-1_6
2. Abraham, A., Andriatsimandefitra, R., Brunelat, A., Lalande, J.F., Viet Triem Tong, V.: GroddDroid: A gorilla for triggering malicious behaviors. In: 2015 10th International Conference on Malicious and Unwanted Software, MALWARE 2015. pp. 119–127. IEEE Computer Society, Fajardo, Puerto Rico (oct 2016). https://doi.org/10.1109/MALWARE.2015.7413692
3. Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: AndroZoo: collecting millions of Android apps for the research community. In: 13th International Workshop on Mining Software Repositories. pp. 468–471. ACM Press, Austin, USA (may 2016). https://doi.org/10.1145/2901739.2903508
4. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P.: FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In: ACM SIGPLAN Conference on Programming Language Design and Implementation. vol. 49, pp. 259–269. ACM Press, Edinburgh, UK (jun 2014). https://doi.org/10.1145/2666356.2594299
5. Chen, K., Wang, X., Chen, Y., Wang, P., Lee, Y., Wang, X., Wang, A., Zhang, Y., Zou, W.: Following Devil's Footprints: Cross-Platform Analysis of Potentially Harmful Libraries on Android and iOS . S&P (2016). https://doi.org/10.1109/SP.2016.29

6. Duan, Y., Zhang, M., Bhaskar, A.V., Yin, H., Pan, X., Li, T., Wang, X., Wang, X.: Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation. In: 24th Annual Network and Distributed System Security Symposium. No. February (2018)
7. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: 9th USENIX Symposium on Operating Systems Design and Implementation. pp. 393–407. USENIX Association, Vancouver, BC, Canada (oct 2010)
8. Enck, W., Ongtang, M., Mcdaniel, P.: On Lightweight Mobile Phone Application Certification Categories and Subject Descriptors (2009)
9. Kiss, N., Lalande, J.F., Leslous, M., Viet Triem Tong, V.: Kharon dataset: Android malware under a microscope. In: The LASER Workshop: Learning from Authoritative Security Experiment Results. pp. 1–12. USENIX Association, San Jose, United States (may 2016)
10. Lalande, J.F., Viêt Triem Tong, V., Leslous, M., Graux, P.: Challenges for Reliable and Large Scale Evaluation of Android Malware Analysis. In: International Workshop on Security and High Performance Computing Systems. pp. 1068–1070. IEEE Computer Society, Orléans, France (jul 2018). https://doi.org/10.1109/HPCS.2018.00173
11. Leslous, M., Viet Triem Tong, V., Lalande, J.F., Genet, T.: GPFinder: Tracking the Invisible in Android Malware. In: 12th International Conference on Malicious and Unwanted Software. pp. 39–46. IEEE Conputer Society, Fajardo (oct 2017). https://doi.org/10.1109/MALWARE.2017.8323955
12. Li, L., Meng, G., Klein, J., Malek, S. (eds.): 1st International Workshop on Advances in Mobile App Analysis, A-Mobile@ASE 2018, Montpellier, France, September 4, 2018. ACM Press (2018). https://doi.org/10.1145/3243218
13. Tam, K., Khan, S., Fattori, A., Cavallaro, L.: CopperDroid: Automatic Reconstruction of Android Malware Behaviors. In: 22nd Annual Network and Distributed System Security Symposium. The Internet Society, San Diego, California, USA (feb 2015)
14. Wei, F., Li, Y., Roy, S., Ou, X., Zhou, W.: Deep ground truth analysis of current android malware. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 10327 LNCS, pp. 252–276. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60876-1_12
15. Wong, M.Y., Lie, D.: IntelliDroid: A Targeted Input Generator for the Dynamic Analysis of Android Malware. In: The Network and Distributed System Security Symposium. pp. 21–24. No. February, The Internet Society, San Diego, USA (feb 2016). https://doi.org/10.14722/ndss.2016.23118
16. Yang, W., Kong, D., Xie, T., Gunter, C.A.: Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In: 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, December 4-8, 2017. pp. 288–302 (2017). https://doi.org/10.1145/3134600.3134642
17. Zhou, Y., Jiang, X.: Dissecting Android Malware: Characterization and Evolution. In: IEEE Symposium on Security and Privacy. pp. 95–109. No. 4, IEEE Computer Society, San Jose, USA (may 2012). https://doi.org/10.1109/SP.2012.16