Security of Android apps

Olga Gadyatskaya

University of Luxembourg
`olga.gadyatskaya@uni.lu`

Thesis Details

Android is a booming eco-system with zillions of third-party apps, many app markets, various devices and multiple platform versions. With a high probability you yourself own (at least) one Android device. Why do not spend your Master or Bachelor thesis pondering about Android security, and trying to improve it?

Below are some tentative thesis topics. In the SaToSS group, we have everything you need to start investigating Android: app datasets, devices, and tools. Android is not as complex as you might think of it. Usually, you will need to learn to install apps on a device/emulator, you will need to run and write relatively small Python programs, and you will need to understand some basic Machine Learning approaches.

Topic 1: Resource-based repackaging detection

Android apps are sources of revenue for their developers, yet it is very easy to plagiarize a third-party app by *repackaging* it. In this thesis you will design a new scheme for detecting repackaged Android apps by using resource files included in the packages. Resource files, such as images, strings, xml layouts, have shown their potential in detecting cloned apps [6]. Subsequent experiments have shown that particular resource file types can serve as better indicators of repackaging. This study, a result of a previous successful Master project, is published in [2]. In your thesis, you will focus on further improvements of the method. The improvements can be in the direction of *robustness* (currently it is very easy for the adversaries to slightly modify the resource files so that the method does not recognize them as identical); *scalability* (improving the performance by moving from pair-wise app comparison to search of the nearest neighbours in some ordered space); or you may focus on developing a *hybrid approach* that will fuse the resource-based detection with some code-based repackaging detection technique.

Details: 60% of time will be dedicated to research, and 40% to development of a prototype to validate the proposed approach. These figures are tentative, and can be further revised depending on how the research will develop.

Topic 2: Dataset building

One of the most challenging tasks in doing Android security is to collect the right dataset to validate the developed approach. In your thesis you will work on collecting a dataset of third-party apps to share with the community. The dataset will be focused on a particular task: *repackaged app detection* (a set of confirmed repackaged and non-repackaged app pairs [2]); *evolution of Android apps* (we want to collect many last-generation apps and check how do they cope with the recent changes in the Android platform architecture [5]); or *malware detection* (a representative set of recent malware samples). Dataset collection typically involves crawling apps from app markets, and querying different online services (e.g., VirusTotal).

Details: tentatively, 50% practical research; 50% development and querying online services.

Topic 3: App code analysis for anomaly detection

This thesis will focus on applying static and dynamic analysis tools (e.g., [4]) to Android apps in order to detect anomalies (e.g., malicious or buggy behaviors). Some theoretical work can also be considered (developing of a semantic model of Android apps expressed as a graph or a state machine).

Details: tentatively, 50% research, 50% development and experiments.

Topic 4: Exploring dynamic app models for malware and repackaging detection

The available scientific literature on malware and repackaging detection often utilizes statically constructed app models (different graphs or graph-derived features, or even code n-grams). However, obfuscation and dynamic code loading hinder application of statically-constructed models (not all code is available at the analysis time) [4]. In this thesis you will explore whether dynamically built models could be successfully applied for Android malware and repackaging detection. You will design a dynamic app model (e.g., a graph), will assess its performance, and will investigate the cost-benefit analysis of using such models (as they will require more time to create, and they will also be incomplete).

Details: tentatively, 40% research, 60% development and experiments.

Topic 5: App runtime performance assessment

Many research tools introduce runtime overhead due to addition of ancillary code (e.g., our own tools for code coverage measurement [7, 3] modify app code), but there is currently no good way to quantify the impact of such tools on runtime performance. You will investigate the Android SDK profiler and other available approaches to measure performance of Android apps and will assess performance penalties of different tools.

Topic 6: Code coverage for automated testing

Code coverage is an important metric for evaluating how well an application has been tested. This is especially critical for third-party Android apps being analyzed for security and reliability purposes. Our group has recently developed ACVTool [3], a new tool to measure code coverage in black-box app testing, and we have many ideas how to extend the tool and how to apply it in the automated testing pipelines. In this thesis project you can focus on 1) extending ACVTool by adding more coverage metrics; or 2) on using ACVTool in an automated testing pipeline (like in [1]) and dynamically improving the testing results by considering the coverage achieved so far.

Details: tentatively, 30% research, 70% development and experiments.

References

1. Dashevskyi, S., Gadyatskaya, O., Pilgun, A., Zhauniarovich, Y.: The influence of code coverage metrics on automated testing efficiency in android. In: Proc. of CCS. pp. 2216–2218. ACM (2018)
2. Gadyatskaya, O., Lezza, A.L., Zhauniarovich, Y.: Evaluation of resource-based app repackaging detection in Android. In: Proc. of NordSec. Springer (2016)
3. Pilgun, A., Gadyatskaya, O., Dashevskyi, S., Zhauniarovich, Y., Kushniarou, A.: An effective android code coverage tool. In: Proc. of CCS. pp. 2189–2191. ACM (2018)
4. Zhauniarovich, Y., Ahmad, M., Gadyatskaya, O., Crispo, B., Massacci, F.: StaDynA: Addressing the problem of dynamic code updates in the security analysis of Android applications. In: Proc. of CODASPY. ACM (2015)
5. Zhauniarovich, Y., Gadyatskaya, O.: Small changes, big changes: An updated view on the Android permission system. In: Proc. of RAID. pp. 346–367. Springer (2016)
6. Zhauniarovich, Y., Gadyatskaya, O., Crispo, B., Spina, F.L., Moser, E.: FSquaDRA: Fast detection of repackaged applications. In: Proc. of DBSec. LNCS, vol. 8566, pp. 130–145. Springer (2014)
7. Zhauniarovich, Y., Philippov, A., Gadyatskaya, O., Crispo, B., Massacci, F.: Towards black-box testing of Android apps. In: Proc. of Software Assurance Workshop at ARES. IEEE (2015)