Thesis Chapter 1: Introduction

Wesley van der Lee

October 2017

Draft version: 2, ext. reviews: F

1 Introduction

Mobile applications play an ever-more important role in our lives. They are the gateways to use social media, to perform banking transactions, to submit working hours and much more. The same applications are not only limited to run on mobile phones, but can also run on other smart devices such as smart-televisions, -watches and -cars. This is in particular true for applications that run on Android, since it is the most popular operating system for smart devices[1]. Although the platform is well established, the security of its applications is not. This locus especially became true when last year researchers of the Norwegian firm Promon were able to exploit Tesla's Android application and achieve full control of the Tesla car paired with the application[1]. A few months after, the research institute Fraunhofer SIT found exploitable vulnerabilities in 9 popular password managers for Android that could compromise the ~~stored~~ passwords by the user [2]. These episodes show that while today's society moves towards a pervasive adoption of mobile applications, these lack sufficient security.

?

stored

currently

The reason why applications deficit security is because software security in general always has been part of a tradeoff where development methodology and time-to-market play an essential role. An early time-to-market generates an early revenue, whereas a security mature development methodology, such as test driven development, results in a more secure application, but a slower overall delivery time.

Modern security testing tools aid the process of discovering bugs by applying a multitude of automated testing techniques that comes in three flavors: black-box, white-box and grey-box testing. White-box testing tests the application's internal logic by code reviews or specific tests. Grey-box testing tests the software's logic using metadata, such as documentation or file structure. Black-box testing treats the software as an interactive application and determines whether the correct output is given for the correct input.

Very fast from mobile security to black-box testing

Black-box testing techniques can also be fine-tuned to understand the application's internal logic. This is what a state machine learning algorithm does, by observing a large number of traces: a combination of inputs and outputs. The inferred state machine reveals a lot of information about the application's logic, and might as such function as a data source for identifying bugs or vulnerabilities in the application. This has been achieved for a wide range of software systems, such as various TLS driver implementations [3]. The inferred models were assessed for extraneous transitions or states and paths that could bypass security measures were found. Another way to use state machine learning is to develop a specification for an implementation when there exists none. This has been performed for the Dutch biometric passport, where it was more time efficient to infer a model through active learning than having a team of experts establish such a model [4].

explain active learning, include a figure.

Model learning can be done on a set of existing traces (*passive learning*) or generate the set of traces while learning by interacting with an application (*active learning*). The drawback of passive learning is that the model is complete in the variety of existing traces, i.e. when certain application behavior is not described in the set of traces, the inferred model describes a deficiency ~~in~~ behavior as well. Active learning overcomes this problem by querying for the information it needs to know. Since software systems, and as a subset to that group Android applications, are able to respond interactively, they function as a qualified data source to generate these traces. The drawback of active learning in this situation is that interacting with an application consumes time, as each input and output combination needs to be simulated. In order to improve the learning process, different active learning algorithms have been developed.

well, the model is incomplete but does not describe this incompleteness!

Because the model that is inferred by active state machine learning manifests additional information about the application, the model might also be used as a new data source to assess the application's security. Up until now, most research involving active learning stops when the model is automatically inferred and continues with manual model inspection for the search of extraneous behavior. The security assessment that is concluded from the manual inspection thus also depends on the reviewer and may yield different results for different reviewers. Furthermore research on retrieving such a model for mobile Android applications is very limited, as there is only one study performed by Lampe et al. that developed a tool for a specific application to infer a state machine model [5].

in fact

it is not clear how this works. –

The aforementioned research provides a limited framework for model inference of mobile applications. Due to its limitations, the framework can function as a basis for this research to first of all overcome these limitations and therefore utilize the inferred model as a data source for an automated security assessment.

1.1 Outline

This thesis conducts research on how to infer a correct state machine model for a generic mobile Android application and assess its security in an automated way. To infer a correct model one has to review active automata learning algorithms and solve problems ~~like~~ the equivalence approximation between a model and an application. Furthermore, algorithms that identify vulnerabilities on the input of an inferred model need to be established. The primary goal of this thesis is

such as

[1] https://developer.android.com/about/android.html

to utilize these building blocks to answer the following main research question of this thesis:

How can one identify weaknesses in mobile Android applications through feasible behavioral state machine learning?

To aid the process of answering the main question, the question has been divided into the following sub-questions:

RQ 1. How can one extend model learning to be applicable to mobile Android applications?
This research question mainly focusses on reducing or mitigating the limitations of the framework provided by Lampe et al. This question deals with the issues that arise when introducing active learning to the mobile application domain, such as the definition of a complete alphabet and identification of mobile parameters.

RQ 2. How can we improve the feasibility of model learning of Android applications? As it has already been mentioned, active learning from simulations is time consuming. Different active learning algorithms reduce the time complexity by limiting the number of queries and the overall query length. This question focuses on the different learning algorithms and corresponding attributes such as model equivalence approximation.

what is ANDROID specific?

RQ 3. How can the learned model be used to assess the application's security? The novelty of this thesis lies in the application of active learning on Android applications and the automatic processing of the model as a new data source to assess the application's security. The latter inquires a set of identification algorithms that determine the presence of a certain vulnerability on input of the inferred model.

-> VERY COOL!

put expectations?

The structure of this thesis is as follows. Chapter 2 gives an overview of the building blocks of active model inference, where various active learning algorithms are discussed as well as techniques to assure model conformance. Chapter 3 reviews the prior work from Lampe et al. by elaborating on the framework they proposed and identifying its limitations. Chapter 4 establishes requirements to overcome the identified limitations and proposes a solution framework that is developed based on these requirements. Chapter 5 establishes algorithms that identify security vulnerabilities in the learned models. Chapter 6 depicts the results of the residual proof of concept running on various mobile Android applications. Chapter 7 discusses these results, answers the research questions and provides a perspective ahead on future work references. This thesis closes by the conclusion stated in Chapter 8.

References

[1] Lars Lunde Birkeland. Tesla cars can be stolen by hacking the app. `https://promon.co/blog/tesla-cars-can-be-stolen-by-hacking-the-app/`, 2016.

[2] Steven Arzt Stephan Huber, Siegfried Rasthofer. Extracting all your secrets: Vulnerabilities in android password managers, 2017.

[3] Joeri De Ruiter and Erik Poll. Protocol State Fuzzing of TLS Implementations. In *USENIX Security Symposium*, pages 193–206, 2015.

[4] Fides Aarts, Julien Schmaltz, and Frits Vaandrager. Inference and abstraction of the biometric passport. *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 673–686, 2010.

[5] KQ Lampe, JCM Kraaijeveld, and TD Den Braber. Mobile application security: An assessment of bunq's financial app. *Delft University of Technology Research Repository*, 2015.