iOS Application Security

ACRONYM	IOSSEC
PROPOSED BY	SCRT Information Security
STUDENT-S	Vladimir Meier
PROFESSOR-S	Jean-Roland Schuler & Rudolf Scheurer
EXPERT-S	Luca Haab
No	B18I14
TYPE	Bachelor Thesis
CONTACT	vladimir.meier@hotmail.com

Context

A majority of apps among the popular ones manipulate sensitive data or expose remotely accessible features. Bugs or misuse of features provided by the operating system puts the user at risk of impersonation, loss of money, data theft, etc.

Consequently, the need to verify that a given functionality is implemented correctly within an application becomes more and more prevalent.

Contributions

The contribution of this work focuses on three aspects of security audits. The first is a testing guide that compiles many of the most common issues in iOS apps. The second revolves around the need to identify and understand algorithms within closed-source software. The third relates to jailbreak detection.

Testing Guide

Finding vulnerabilities in software requires both a methodology that does not depend on the underlying technology and the ability to recognize a wide range of patterns of software issues. While the former is trained by experience, the latter can be quickly digested if the right material is available. As such, the proposed Testing Guide allows a security engineer to conduct a security assessment with minimal preexisting knowledge on iOS.

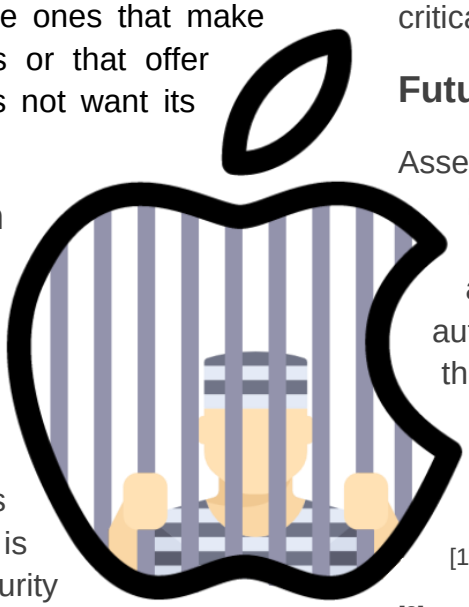
Reverse Engineering

When performing software security assessments, having the source code is always a plus. In spite of that, the hazards of the profession often require to understand or modify behavior in closed-source systems. Consequently, a black-box testing guide is proposed as well. It delves into reverse engineering of iOS applications and make them spit meaningful information.

Jailbreak

On iOS, “jailbreaking” roughly consists in disabling code signing verification. Users would come to that for mainly two reasons:

1. An iOS device refuses to install unsigned or untrusted applications, meaning an iPhone’s buyer is not allowed to install whatever it would like on its smartphone.
2. Apple reviews the applications submitted by the developers before publishing them on the App Store. The company does not hesitate to reject the ones that make use of private APIs or that offer features Apple does not want its devices to possess.



Jailbreak Detection

More and more applications attempt to determine whether they run on a jailbroken device and block access to features if that is the case. This is a problem for security engineers, as they rely on tools that perform privileged actions unavailable on factory devices.

To tackle this problem, a study of state-of-the-art jailbreak detection techniques precedes a methodology to defeat such protections, with a practical example on a popular banking application. Afterwards, a proof-of-concept of automatic jailbreak detection bypass is realized using runtime instrumentation of platform APIs.

Evaluation

The Testing Guide is applied on a famous password manager and leads to the discovery of a dozen of issues, ranging from benign to critical.

Future Works

Assessing a software’s security involves many repetitive tasks. In addition, software weaknesses often manifest themselves in a recognizable manner. As such, automated binary analysis may help improve the scalability of vulnerability detection in iOS applications.

Sources

[1] iPad: https://www.apple.com

[2] pentest: https://agsnig.com/penetration-testing.html