TARSKI TECHNOLOGIES

REVEALING TRUTH WITH DATA-DRIVEN EXPERIENCES™

WALLAWALLET SECURITY REPORT

PREPARED FOR
WALLAWALLET

NOV 30, 2020

WWW.TARSKI.TECH

WALLAWALLET IOS SECURITY REPORT

This document provides a secure code review of the Wallawallet mobile H-Bar wallet, as defined below in the "Scope" section. The results of the audit are covered in this document.

I. INTRODUCTION

Aidan Noel, contracting through Tarski Technologies LLC, has conducted a secure code review of the Wallawallet mobile H-Bar wallet, as defined below in "Scope". The results of the audit are covered in this document. The code audit began the 11th of November 2020 and lasted until the 30th of November 2020. The objective of this assessment was to determine security weaknesses in the mobile application, with a focus on secure key management, cryptographic standards and the implementations of cryptographic functions.

It should be noted that, while the review process will be as thorough as possible in discovering and reporting security vulnerabilities, it is not always guaranteed to find all possible security weaknesses. If no issues are found, this review does not certify that the application is 100% invulnerable to attacks. A third-party secure code review should be considered a strong addition to a company's overarching risk mitigation program, but not a "silver bullet" in the continuous risk mitigation process.

II. BACKGROUND

Wallawallet is a mobile cryptocurrency wallet developed by Ledgerama, LLC. The application is designed to manage accounts storing H-Bar, the cryptocurrency utilized in Hedera Hashgraph transactions. As such, the security of the application is key to guaranteeing the safety of users' personal financial assets. This audit covered the pieces of code unique to the iOS version of the application, as I have previously completed an audit of the Android version. The application is currently pending review for access on the Apple App Store.

III. SCOPE

A secure code review was performed on the Wallawallet code base on Gitlab. The review included only mobile application code for the iOS version of the application, with a focus on security-essential code unique to this version, as other code common to both the Android and iOS versions was already reviewed in my first audit. Some React Native code previously reviewed in the Android-version audit was also re-investigated.

IV. ASSESSMENT

I. CRYPTOGRAPHY AND KEY MANAGEMENT

As the client's primary concern was secure key management in their application, this was the initial focus while reviewing the codebase. The application adheres to the same security best practices as the Android version, including the 12-byte initialization vectors (IVs) corrected in the first audit. Symmetric encryption and decryption operations are performed with AES Galois/Counter Mode (GCM) and 12-byte IVs. Asymmetric operations are performed with RSA SHA-256 with MGF1 padding. IVs and the 32-byte application key are generated using a suitable CSPRNG.

The iOS version of the application is vulnerable to the same hardware attacks as the Android version, such as RowHammer or other side-channel analysis attacks. These vulnerabilities can be considered low severity due to the difficulty of obtaining a device and executing an attack before the application key timeout occurs, as well as the difficulty of guessing the user's 6 digit pin in the case that it is set.

II. REACT NATIVE SECURITY

The React Native application components separate from the security module were re-investigated for vulnerabilities with a focus on the OWASP Top Ten Application Security Risks.(1) This code was previously reviewed in the Android version audit. This assessment discovered no new vulnerabilities in the general React Native mobile application.

V. CONCLUSION

Recommendations made in the previous audit allowed the iOS version to be implemented without any new vulnerabilities. Physical attack vectors discovered in the iOS version of the Wallawallet application should be considered Low severity, as they require the device to be acquired and attacked while the user is authenticated in the application. This opportunity can be reduced by the user decreasing the timeout limit on the application key and utilizing the additional 6-digit pin feature. Malware based attacks on the application require a malicious application to be installed on the device, or some form of email or browser-based script injection or remote code execution. Modern browsers, email clients, and device security largely prevent attacks of this kind, and mitigation is entirely dependent on a user's security awareness.

VI. FOOTNOTE

1. https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities