Mobile Star Key Wallet Security Audit Report

CDSecurity

Revision: 25 April 2024

Audit Report Composer: Chris

Overview

Background

Star Key Wallet team requested web extension wallet audit to CDSecurity.

Project Dates

2024.4.5 ~ 4.25: Code Review
2024.4.26: Initial Audit Report

Review Team

- Chris : Cryptography Researcher and Engineer, Blockchain Engineer

Coverage

Target Code

The following code repositories are considered in-scope for the review:
https://github.com/madMax92221/Mobile-wallet

Supporting Documentation

No document was provided.

Findings

Specific Issues and Suggestions

Issue 1(High): Weak Access Control for react-native-keychain

Location

src\services\KeyChainService.ts#L10-L12

Descriptions

Star Key Wallet uses the setGenericPassword function from the react-native-keychain library to store the user's password and address in the device. This function has different options, one being that Keychain.ACCESS_CONTROL which has many access control choices that require a password or touch ID to access the keychain data. However, by default, there is no access control while accessing the keychain data

```
L10: const storePasswordInKeyChain = async (password: string) => {
L11:    await Keychain.setGenericPassword(SERVICE_NAME, password);
L12:  };
```

Recommendation & Mitigations

It is recommended to specify the desired access control from available options.

```
const storePasswordInKeyChain = async (password: string) => {
```

```
--    await Keychain.setGenericPassword(SERVICE_NAME, password);
++    await Keychain.setGenericPassword(SERVICE_NAME, password, {
++          accessControl:
Keychain.ACCESS_CONTROL.BIOMETRY_ANY_OR_DEVICE_PASSCODE
++    });

  };
```

Issue 2(High): Wallet Screen Not Protected Against Screen Recording

Location

Android
ios

Synopsis

On Android and some versions of iOS, the user or an application may record videos or individual frames from the Star Key Wallet application.
The data displayed by the Star Key Wallet is critical to both the security of the application and the privacy of the user. This data includes the mnemonic phrase, which is displayed during wallet key pair generation, and at any time after that in the backup screen. A leak of the mnemonic passphrase is a critical breach of the security of the wallet. Additionally, the wallet displays user account balance and transaction history, of which a malicious screenshot could leak private user data.
Private user data must be secured and protected from access by other applications running on the mobile device.

Impact

A malicious application accessing an image of the mnemonic phrase could result in loss of all wallet funds. In addition, account balance or transaction history leaks are a breach of user privacy.

Preconditions

The user must make a screenshot, and grant other applications access to the screenshot. Alternatively, an application that has access to the contents of the screen (e.g. a screen recording application) would be required.

Feasibility

The feasibility of such an attack depends on the specific phone and OS.
On iOS 10 or newer, there is no API for an application to use to access the screen of another application, so this attack vector is not applicable. However, if the attacker plants a malicious application with access to the photo gallery on the device and the user makes a screenshot of the seed phrase or other private information, it is possible for the application to access it.
On Android, the MediaProjection API can be used to record the contents of the screen. The user has to explicitly consent to the access, and during the duration of the screen recording, an icon is shown. This icon represents wireless transmission of the screen contents to a projector or ChromeCast and may not be immediately identifiable as a screen recording icon. Additionally, any application with access to the files of the user can access all screenshots.

Recommendation & Mitigations

I suggest that the Star Key Wallet team take the following measures to prevent or mitigate the impact of screenshots.

Android

On Android, we recommend setting FLAG_SECURE on the application window and making sure that no private content is shown in other windows. For more information, we suggest referring to this blog post about vulnerabilities of weak screenshot protection. The flag can either be set manually or by using an existing React module.

iOS

On iOS, there is no API to prevent screenshots, which makes mitigating this issue more difficult. In some instances, the only thing that can be done is to remind the user that taking screenshots of the seed means making it available to other applications. In other cases, such as the seed generation screen or when the application is being AirPlayed or mirrored, stronger measures could be taken. In this case, the iOS event UIApplicationUserDidTakeScreenshotNotification can be handled to display a notice to the user that the action is not secure, and to generate and display a new seed. With this approach, instead of preventing the screenshotting of a seed that is used, using a seed that is screenshotted is prevented. ScreenShieldKit also warrants mention in this case as a potential resource for preventing screenshots.