Raylac Wallet Code Review

Security Assessment

CertiK Assessed on Jan 31st, 2025

CertiK Assessed on Jan 31st, 2025

Raylac Wallet Code Review

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Platform	Ethereum (ETH)	Dynamic Testing, Manual Review, Testnet Deployment

LANGUAGE	TIMELINE	KEY COMPONENTS
TypeScript	Delivered on 01/31/2025	N/A

Vulnerability Summary

9	7	0	0	2	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed immediately. Users should be cautious when interacting with any application with outstanding critical risks.
0	High		High risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds, thief of user data, and/or loss control of the application.
1	Medium	1 Resolved	Medium risks may not pose a security risk at a large scale, but they can affect the overall functioning of a platform or be used to target a certain group of users.
7	Low	6 Resolved, 1 Acknowledged	Low risks can be any of the above, but on a smaller impact. They generally do not compromise the overall integrity of the project.
1	Informational	1 Acknowledged	Informational errors are often recommendations to improve the configuration or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the application.

TABLE OF CONTENTS | RAYLAC WALLET CODE REVIEW

APPROACH & METHODS | RAYLAC WALLET CODE REVIEW

This report has been prepared for Raylac to discover issues and vulnerabilities in the application of the Raylac Wallet Code Review project. Raylac is a wallet that allows users to store, send, swap, and receive Ethereum and ERC-20 tokens.

The pentest was a manual assessment of the security of the application's functionality, business logic, and vulnerabilities, such as those cataloged in the OWASP Top 10. The assessment also included a review of security controls and requirements listed in the OWASP Application Security Verification Standard (ASVS). The pentesters leveraged tools to facilitate their work. However, the majority of the assessment involved manual analysis.

The main objective of the engagement is to test the overall resiliency of the application to various real-world attacks against the application's controls and functions and thereby be able to identify its weaknesses and provide recommendations to fix and improve its overall security posture.

Two members of the CertiK team completed the engagement over two days in January 2025, identifying 9 security-relevant findings. Weaknesses were found and are detailed in the Findings section of the report. We recommend addressing these findings to maintain high security standards, align with industry best practices, and enhance the application's security posture.

REVIEW NOTES | RAYLAC WALLET CODE REVIEW

The scope of this review was limited to the code related to key management. Additionally, these functionalities were reviewed in the mobile applications (iOS and Android) under development. Below is a list of the files included in the scope.

- ShowMnemonicSheet.tsx
- useCreateAccount.ts
- useDeriveAddress.ts
- useImportMnemonic.ts
- useMnemonic.ts
- key.ts
- ConfirmBackupPhrase.tsx
- CreateAddress.tsx
- ImportAccount.tsx
- PrivateKeyAddressDetailsSheet.tsx
- SaveBackupPhrase.tsx
- useSwap.ts
- useSend.ts
- useBridgeSend.ts
- useLoadPrivateKey.ts

FINDINGS | RAYLAC WALLET CODE REVIEW

9	0	0	1	7	1
Total Findings	Critical	High	Medium	Low	Informational

This report has been prepared to discover issues and vulnerabilities for Raylac Wallet Code Review. Through this security assessment, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Dynamic Testing, Manual Review & Testnet Deployment to complement rigorous testing process, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-06	Lack Of Local Authentication	Account Policy	Medium	Resolved
GLOBAL-01	Screenshot Backgrounding	Information Disclosure	Low	Resolved
GLOBAL-02	Sensitive Information Leaking In Memory	Information Disclosure	Low	Resolved
GLOBAL-03	Insufficient Secure Keystore Protection	Insecure Data Storage	Low	Resolved
GLOBAL-04	Mnemonic Display Allow Screenshot	Security Misconfiguration	Low	Resolved
GLOBAL-05	Mnemonic And Private Key Are Copied To Clipboard	Information Disclosure	Low	Acknowledged
GLOBAL-07	Custom Keyboards Allowed For Sensitive Inputs	Information Disclosure	Low	Resolved
GLOBAL-08	The `require_authtentication` Option Depends On The Device	Security Misconfiguration	Low	Resolved
GLOBAL-09	Code Quality Improvements	coding	Informational	Acknowledged

GLOBAL-06 | LACK OF LOCAL AUTHENTICATION

Category	Severity	Location	Status
Account Policy	Medium		Resolved

Introduction

Local authentication refers to the process of an application authenticates the user against credentials stored locally on the device. The user "unlocks" the application or some inner layer of functionality by providing a valid PIN, password or biometric characteristics such as face or fingerprint, which is verified by referencing local data. Generally, this is done so that users can more conveniently resume an existing session with a remote service or as a means of step-up authentication to protect some critical function within the applications.

Description

The mobile applications do not require users to enter a password/PIN/Biometric when opening the application. Additionally, the Android application does not require authentication to access the private key.

Impact

An attacker can access the victim's wallet secrets if he has physical access to the unlocked device. He can take over the account using the victim's wallet secrets and take away all user's funds.

Recommendation

It is recommended to implement local authentication when launching the application or attempting to access the wallet secrets.

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-01 | SCREENSHOT BACKGROUNDING

Category	Severity	Location	Status
Information Disclosure	Low	Android application iOS application	Resolved

Description

On mobile devices, a screenshot of the current activity is typically taken when an application moves to the background and displayed for aesthetic purposes when the app returns to the foreground. This feature can pose a security risk, as sensitive data may be exposed if the user backgrounds the application while sensitive information is displayed. Additionally, a malicious application running on the device and capable of continuously capturing the screen could also expose sensitive data.

Impact

An attacker with physical access to an unlocked device or a malicious third-party app with access to the auto-generated screenshot of the application could retrieve sensitive information contained in the screenshot.

Recommendation

It is recommended for the application to add an overlay to hide or obscure the application screen before it moves to the background.

- **iOS**: Add an overlay screen before the application transitions to the background, and remove it when the application returns to the foreground.
- **Android**: In addition to adding an overlay, enable the `FLAG_SECURE` option. The `FLAG_SECURE` flag helps prevent sensitive information from being included in the auto-generated screenshot.

For more information about the FLAG_SECURE flag:

- https://developer.android.com/reference/android/view/Display#FLAG_SECURE
- https://stackoverflow.com/questions/9822076/how-do-i-prevent-android-taking-a-screenshot-when-my-app-goes-to-the-background

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-02 | SENSITIVE INFORMATION LEAKING IN MEMORY

Category	Severity	Location	Status
Information Disclosure	Low	Android application iOS application	Resolved

Description

Sensitive information leaking in memory happens when an application handles confidential data improperly, causing it to remain exposed in the system's memory. This can happen if an application or service fails to securely clear this sensitive information after use, or if the information is stored insecurely within memory. This could allow attackers to directly read the memory contents if they gain access to the device.

The source code implements mechanisms to reset states such as the `clearState` function and `useEffect`, but they are not applied in all cases. For example:

- packages/app/src/screens/ImportAccount/PrivateKeyAddressDetailsSheet.tsx:82
- packages/app/src/screens/SaveBackupPhrase/SaveBackupPhrase.tsx

It is recommended to review the entire codebase.

Impact

Attackers can gain access to sensitive data stored in memory such as mnemonic and private keys if the application does not properly secure it.

Reproduce Steps

1. any functionality that uses the private key or mnemonic. For example, show private key.
2. Navigate through the application, put it in the background, and then reopen it.
3. Dump the memory with `fridadump3`. This tool creates the `dump` directory with the `strings.txt` file along with multiple `.data` files.

```
python3 fridump3.py -s Raylac
```

Proof of Concept

The screenshot below demonstrates that the Mnemonic and private key is visible in the memory dump.

Recommendation

It is recommended the following actions:

- Ensure that all sensitive information stored in memory is encrypted using strong cryptographic algorithms. This minimizes the risk of data being readable if memory is accessed by unauthorized users or attackers.
- Use secure memory management techniques to ensure that sensitive data is wiped from memory once it is no longer needed. This can include zeroing out memory buffers after use and using memory protection features available in the operating system or development environment.
- Do not use immutable structures (e.g., String and BigInteger)
- Reassign values to the variables used to store sensitive data.

Reference

- https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d-Testing-Data-Storage.md#checking-memory-for-sensitive-data-mstg-storage-10
- https://books.nowsecure.com/secure-mobile-development/en/coding-practices/securely-store-sensitive-data-in-ram.html
- https://developer.android.com/training/articles/security-tips#UserData

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-03 | INSUFFICIENT SECURE KEYSTORE PROTECTION

Category	Severity	Location	Status
Insecure Data Storage	Low	Android application	Resolved

Description

The Android application is designed to create a cryptographic key from the Keystore. This key is intended for encrypting the wallet's mnemonic and private key to ensure secure storage within the application's shared preferences. However, the Keystore entries currently lack sufficient protection, as several security flags are not enabled:

- `isTrustedUserPresenceRequired`
- `isUserAuthenticationRequired`
- `isUserAuthenticationRequirementEnforcedBySecureHardware`
- `isUserConfirmationRequired`

These flags are for ensuring that the keystore entries are protected against unauthorized access.

Impact

The absence of these security protections leaves the keystore entries vulnerable to unauthorized access. An attacker who gains access to the device might be able to retrieve sensitive cryptographic keys, thereby compromising the confidentiality and integrity of the encrypted wallet mnemonics.

Proof of Concept

1. Open the application
2. Run the `tracer-keystore.js` script for `frida` : https://raw.githubusercontent.com/WithSecureLabs/android-keystore-audit/master/frida-scripts/tracer-keystore.js
3. List the keystore entries with the `ListAliasesAndroid()` function
4. Check the information of the entries with the `AliasInfo('<entrie>'')` function

```
[Keystore.getKey()]: alias: AES/GCM/NoPadding:key_v1:keystoreUnauthenticated, password: '(null)'
{
    "blockModes": [
        "GCM"
    ],
    "digests": [],
    "encryptionPaddings": [
        "NoPadding"
    ],
    "isInsideSecureHardware": true,
    "isInvalidatedByBiometricEnrollment": false,
    "isTrustedUserPresenceRequired": false,
    "isUserAuthenticationRequired": false,
    "isUserAuthenticationRequirementEnforcedBySecureHardware": false,
    "isUserAuthenticationValidWhileOnBody": false,
    "isUserConfirmationRequired": false,
    "keyAlgorithm": "AES",
    "keySize": 256,
    "keyValidityForConsumptionEnd": null,
    "keyValidityForOriginationEnd": null,
    "keyValidityStart": null,
    "keystoreAlias": "AES/GCM/NoPadding:key_v1:keystoreUnauthenticated",
    "origin": 1,
    "purposes": 3,
    "signaturePaddings": [],
    "userAuthenticationValidityDurationSeconds": -1
}
[Remote::com.raylac.app ]->
```

Recommendation

It is recommended the following actions to improve the security of the keystore entries:

- Enforce user authentication: Update the keystore entries to require user authentication before granting access to the sensitive information stored within them. Set the `isUserAuthenticationRequired` flag to true to enforce this requirement.
- Leverage hardware-backed security: Ensure that the `isUserAuthenticationRequirementEnforcedBySecureHardware` flag is set to true, which means that user authentication is enforced by the secure hardware, providing an additional layer of security.
- Implement user presence verification: Enable the `isTrustedUserPresenceRequired` flag to ensure that the user is physically present and actively verifying their identity when accessing the sensitive information stored in the keystore entries.
- Require user confirmation: Set the `isUserConfirmationRequired` flag to true to ensure that users must explicitly confirm their intention to perform sensitive operations, such as signing a transaction, before the application proceeds.

Note that although the examples are given on Android Keystore system, similar protections are also presented on iOS secure enclave/keychain system.

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-04 | MNEMONIC DISPLAY ALLOW SCREENSHOT

Category	Severity	Location	Status
Security Misconfiguration	Low	Android application iOS application	Resolved

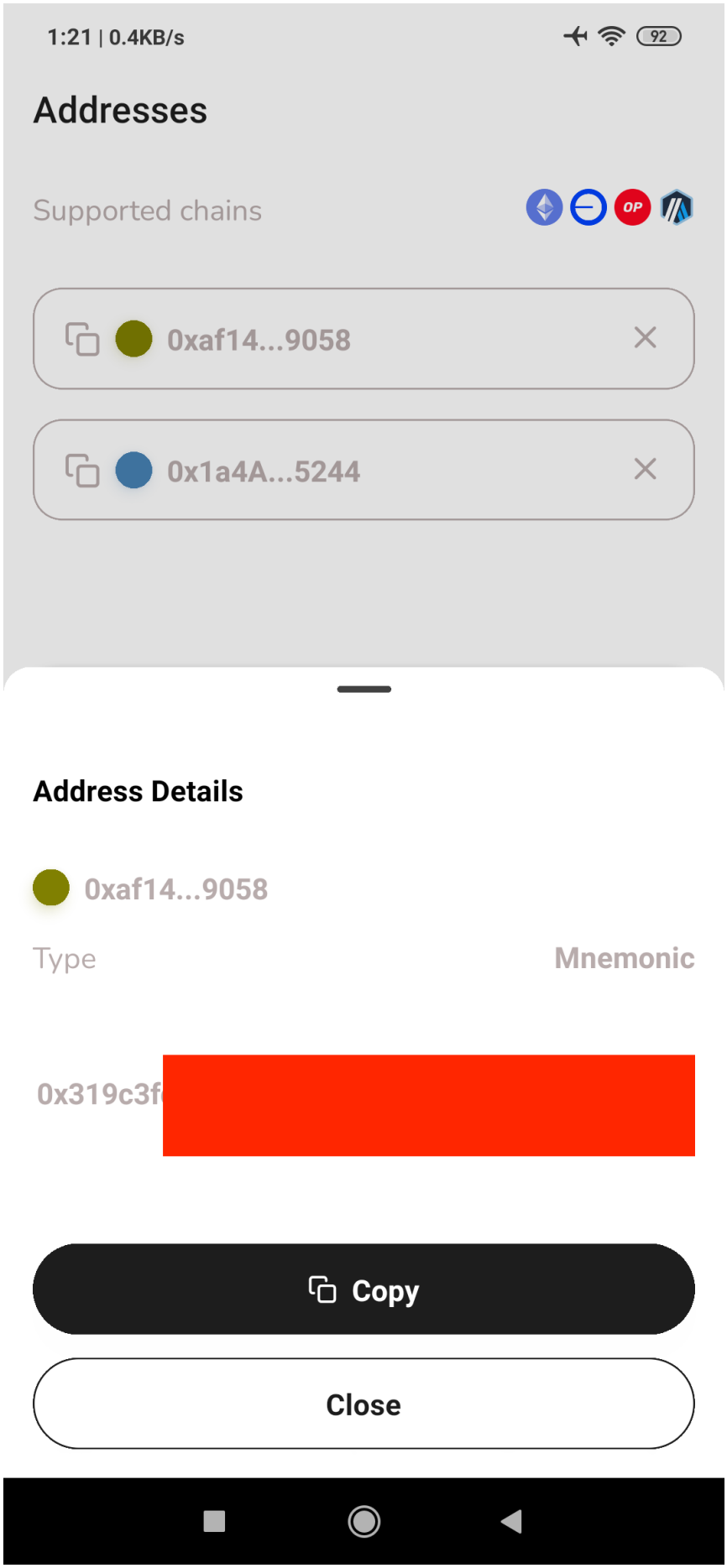
Description

The application does not have a mechanism to prevent users from taking screenshots of the displayed wallet secrets, nor does it display a warning to remind users of the risks associated with taking screenshots. Obtaining the mnemonic or private key could potentially allow an attacker to gain full control of the wallet.

Impact

Third-party apps with the "READ_EXTERNAL_STORAGE" permission on an Android device or apps with full photo access on an iPhone can access screenshots stored on the device. These apps could potentially retrieve the mnemonic if it or the private key is included in a screenshot taken by the user.

Proof of Concept


1:21 | 0.4KB/s
92
Addresses
Supported chains
OP
0xaf14...9058
0x1a4A...5244
Address Details
0xaf14...9058
Type
Mnemonic
0x319c3f
Copy
Close

Recommendation

Android

Screen capture can be prevented by enabling the `FLAG_SECURE` option. The `FLAG_SECURE` flag helps prevent users and malicious third-party apps from recording the mnemonic screens or taking screenshots of sensitive information.

For more information about the `FLAG_SECURE` flag, please see:
https://developer.android.com/reference/android/view/Display#FLAG_SECURE

iOS

There is no built-in solution on iOS to prevent users from taking screenshots. It is recommended to add a warning to remind users not to take screenshots when viewing their wallet secrets.

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-05 MNEMONIC AND PRIVATE KEY ARE COPIED TO CLIPBOARD

Category	Severity	Location	Status
Information Disclosure	Low		Acknowledged

Description

The application allows the mnemonic phrase and private key to be copied to the clipboard. The clipboard is a shared system resource that can be accessed by other applications running on the device. This means that any malicious application or malware with clipboard access can potentially read and extract the sensitive data, leading to unauthorized access to the user's wallet.

The feature is implemented in the `packages/app/src/lib/utils.ts` file and called from these files:

- `packages/app/src/components/AddressDetailsSheet/PrivateKeyAddressDetailsSheet.tsx`
- `packages/app/src/components/ShowMnemonicSheet/ShowMnemonicSheet.tsx`
- `packages/app/src/screens/SaveBackupPhrase/SaveBackupPhrase.tsx`

There are more files out of scope that call the `copyToClipboard` function.

Impact

Copying the mnemonic phrase or private key to the clipboard increases the risk of credential theft. If a malicious app running in the background monitors clipboard activity, it can capture the copied mnemonic or private key, allowing an attacker to gain full control over the user's wallet. Additionally, some operating systems and applications may store clipboard history, increasing the risk of long-term exposure. Users who unintentionally paste their keys into insecure locations further compound the risk, leading to potential loss of funds and unauthorized transactions.

Recommendation

Consider not allowing users to copy the mnemonic and private key to the clipboard. Ensure they store the phrase by requiring them to retype it in a subsequent view during onboarding and backup.

Alleviation

[Raylac Team, 02/05/2025]: The team acknowledged the finding and decided not to change the current codebase.

GLOBAL-07 CUSTOM KEYBOARDS ALLOWED FOR SENSITIVE INPUTS

Category	Severity	Location	Status
Information Disclosure	Low	Android application iOS application	Resolved

Description

A custom keyboard is a keyboard app developed by a third-party company or developer other than the device's manufacturer. These keyboards offer users the flexibility to enhance functionality and use different themes. The mobile applications do not disable custom keyboards when entering wallet mnemonics. Users can install custom keyboards that replace the system's default keyboard in any app. These custom keyboards can log and exfiltrate the data users enter.

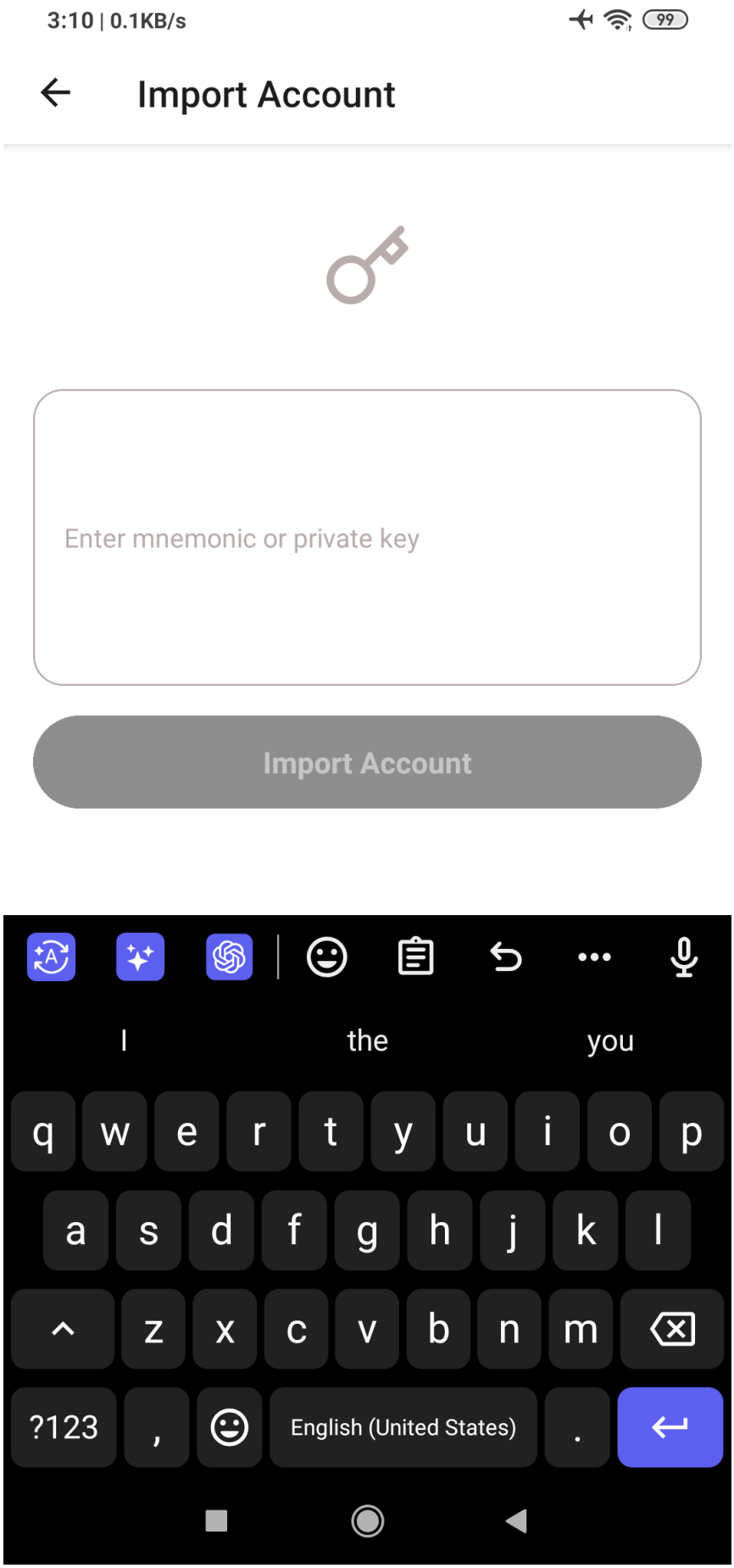
Impact

If a user has installed a custom keyboard that logs and exfiltrates data, sensitive information such as wallet mnemonics or private keys entered using this keyboard can be collected by the third-party keyboard and potentially compromised.

Reproduce Steps

1. Download third-party keyboard from the official store
2. Install it
3. Open the app and import wallet with mnemonic
4. Switch to the third-party keyboard

Proof of Concept


3:10 | 0.1KB/s
99
Import Account
Enter mnemonic or private key
Import Account
I
the
you
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
,
English (United States)
.

Recommendation

It is recommended to disable third-party keyboards within the application to prevent the leakage of sensitive data entered by the user.

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-08 THE `require_authtentication` OPTION DEPENDS ON THE DEVICE

Category	Severity	Location	Status
Security Misconfiguration	Low		Resolved

Description

The application stores the sensitive information with the SecureStore module from `expo-secure-store`. Specifically, it is used the `setItem` and `setItemAsync` functions which receive the parameter `requireAuthentication`. This option is responsible for enabling the usage of the user authentication methods available on the device while accessing data stored in SecureStore.

- Android: Equivalent to setUserAuthenticationRequired(true) (requires API 23).
- iOS: Equivalent to biometryCurrentSet. Complete functionality is unlocked only with a freshly generated key - this would not work in tandem with the keychainService value used for the others non-authenticated operations.

The value of `requireAuthentication` is based on the result of `Device.isDevice` from the `expo-device` package.

- `packages/app/src/lib/key.ts:14`

```
`const REQUIRE_AUTHENTICATION = Device.isDevice;`
```

Since this is a significant security option for protecting the private key and mnemonic of the wallet, it is recommended to always set it to `true`.

Impact

The `Device.isDevice` function simply determines whether the application is running on a physical device rather than an emulator. It does not verify if user authentication is properly enforced, meaning that in certain scenarios, authentication may not be required to access stored sensitive data. Additionally, the function can return an unexpected result. For example: https://github.com/expo/expo/issues/16165

Recommendation

It is recommended to always set `requireAuthentication` to true to ensure that stored sensitive data is accessible only after user authentication. Instead of relying on `Device.isDevice`, explicitly enforce authentication for all SecureStore operations that handle sensitive wallet data.

Alleviation

[Raylac Team, 02/05/2025]: The team heeded the advice and resolved the issue in commit: 8b068bb46305ffa63de2bca454962ff27365a6a1.

GLOBAL-09 | CODE QUALITY IMPROVEMENTS

Category	Severity	Location	Status
coding	Informational		Acknowledged

Description

During the code review, it was identified the following areas for improvement in code quality. These are some examples:

1. Use of multiple libraries with overlapping functionalities. The code uses the `bip39` and `viem` libraries for generating and managing wallets.
2. Multiple calls to the `sleep` function. Excessive use of sleep functions can cause unnecessary delays, impact performance, and introduce race conditions.

- `packages/app/src/hooks/useCreateAccount.ts:37`

3. Redundant conditions. Logical redundancies, such as checking an already boolean variable, can lead to unnecessary processing and reduced code readability.
    - `packages/app/src/hooks/useImportMnemonic.ts:50`. The `imported` variable is already a boolean.
4. Improper error handling. Inconsistent or missing error handling can lead to unhandled exceptions, unexpected crashes, and potential security risks.

- `packages/app/src/hooks/useImportMnemonic.ts:42`

Impact

Poor code quality can introduce security and stability risks, making the application more prone to crashes, performance bottlenecks, and unexpected behavior. Additionally, using multiple libraries for the same functionality increases the attack surface, complicates dependency management, and can lead to outdated or vulnerable packages remaining in use.

Recommendation

It is recommended to optimize the code by removing unnecessary sleep function calls, eliminating redundant conditions, and ensuring proper error handling practices. Consolidate library dependencies by reviewing and selecting a single, well-maintained library for each functionality to reduce redundancy and improve maintainability.

Alleviation

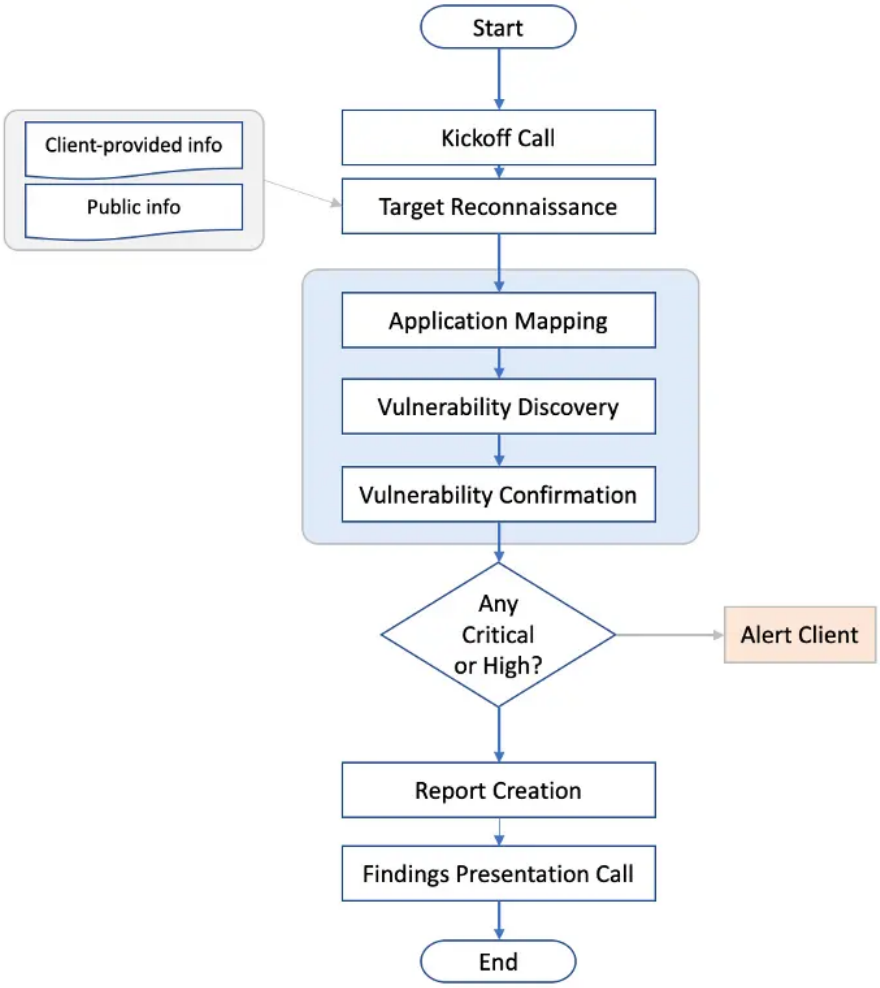
[Raylac Team, 02/05/2025]: The team acknowledged the finding and decided not to change the current codebase.

APPENDIX | RAYLAC WALLET CODE REVIEW

Methodology

CertiK uses a comprehensive penetration testing methodology which adheres to industry best practices and standards in security assessments including from OWASP (Open Web Application Security Project), NIST, PTES (Penetration Testing Execution Standard).

Below is a flowchart of our assessment process:



Coverage and Prioritization

As many components as possible will be tested manually. Priority is generally based on three factors: critical security controls, sensitive data, and the likelihood of vulnerability.

Critical security controls will always receive the top priority in the test. If a vulnerability is discovered in the critical security control, the entire application is likely to be compromised, resulting in a critical-risk to the business. For most applications, critical controls will include the login page, but it could also include major workflows such as the checkout function in an online store.

The Second priority is given to application components that handle sensitive data. This is dependent on business priorities,

but common examples include payment card data, financial data, or authentication credentials.

Final priority includes areas of the application that are most likely to be vulnerable. This is based on CertiK' experience with similar applications developed using the same technology or with other applications that fit the same business role. For example, large applications will often have older sections that are less likely to utilize modern security techniques.

Reconnaissance

CertiK gathers information about the target application from various sources depending on the type of test being performed. CertiK obtains whatever information that is possible and appropriate from the client during scoping and supplements it with relevant information that can be gathered from public sources. This helps provide a better overall picture and understanding of the target.

Application Mapping

CertiK examines the application, reviewing its contents, and mapping out all its functionalities and components. CertiK makes use of different tools and techniques to traverse the entire application and document all input areas and processes. Automated tools are used to scan the application and it is then manually examined for all its parameters and functionalities. With this, CertiK creates and widens the overall attack surface of the target application.

Vulnerability Discovery

Using the information that is gathered, CertiK comes up with various attack vectors to test against the application. CertiK uses a combination of automated tools and manual techniques to identify vulnerabilities and weaknesses. Industry-recognized testing tools will be used, including Burp Suite, Nikto, Metasploit, and Kali. Furthermore, any controls in place that would inhibit the successful exploitation of a particular system will be noted.

Vulnerability Confirmation

After discovering vulnerabilities in the application, CertiK validates the vulnerabilities and assesses its overall impact. To validate, CertiK performs a Proof-of-Concept of an attack on the vulnerability, simulating real world scenarios to prove the risk and overall impact of the vulnerability.

Through CertiK's knowledge and experience on attacks and exploitation techniques, CertiK is able to process all weaknesses and examine how they can be combined to compromise the application. CertiK may use different attack chains, leveraging different weaknesses to escalate and gain a more significant compromise.

To minimize any potential negative impact, vulnerability exploitation was only attempted when it would not adversely affect production applications and systems, and then only to confirm the presence of a specific vulnerability. Any attack with the potential to cause system downtime or seriously impact business continuity was not performed. Vulnerabilities were never exploited to delete or modify data; only read-level access was attempted. If it appeared possible to modify data, this was noted in the list of vulnerabilities below.

Immediate Escalation of High or Critical Findings

If critical or high findings are found whereby application elements are compromised, client's key security contacts will be notified immediately.

Risk Assessment

Risk Level	CVSS Score	Impact	Exploitability
Critical	9.0-10.0	Root-level or full-system compromise, large-scale data breach	Trivial and straightforward
High	7.0-8.9	Elevated privilege access, significant data loss or downtime	Easy, vulnerability details or exploit code are publicly available, but may need additional attack vectors (e.g., social engineering)
Medium	4.0-6.9	Limited access but can still cause loss of tangible assets, which may violate, harm, or impede the org's mission, reputation, or interests.	Difficult, requires a skilled attacker, needs additional attack vectors, attacker must reside on the same network, requires user privileges
Low	0.1-3.9	Very little impact on an org's business	Extremely difficult, requires local or physical system access
Informational	0.0	Discloses information that may be of interest to an attacker.	Not exploitable but rather is a weakness that may be useful to an attacker should a higher risk issue be found that allows for a system exploit

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.