



Wallet Application Security Audit Report



Table Of Contents

1 Executive Summary	_____
2 Audit Methodology	_____
3 Project Overview	_____
3.1 Project Introduction	_____
3.2 Vulnerability Information	_____
3.3 Vulnerability Summary	_____
4 Audit Result	_____
5 Statement	_____

1 Executive Summary

On 2024.08.22, the SlowMist security team received the kucoin team's security audit application for KuCoin Wallet Android, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "black-box and grey-box" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for wallet application includes two steps:

The codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The wallet application is manually analyzed to look for any potential issues.

The following is a list of security audit items considered during an audit:

NO.	Audit Items	Result
1	App runtime environment detection	Passed
2	Code decompilation detection	Passed
3	App permissions detection	Passed
4	File storage security audit	Passed
5	Communication encryption security audit	Passed
6	Interface security audit	Passed
7	Business security audit	Passed
8	WebKit security audit	Passed
9	App cache security audit	Passed
10	WebView DOM security audit	Passed
11	SQLite storage security audit	Passed
12	Deeplinks security audit	Passed
13	Client-Based Authentication Security audit	Passed
14	Signature security audit	Passed
15	Deposit/Transfer security audit	Passed
16	Transaction broadcast security audit	Passed

NO.	Audit Items	Result
17	Secret key generation security audit	Passed
18	Secret key storage security audit	Passed
19	Secret key usage security audit	Passed
20	Secret key backup security audit	Passed
21	Secret key destruction security audit	Passed
22	Screenshot/screen recording detection	Passed
23	Paste copy detection	Passed
24	Keyboard keystroke cache detection	Passed
25	Insecure entropy source audit	Passed
26	Background obfuscation detection	Passed
27	Suspend evoke security audit	Passed
28	AML anti-money laundering security policy detection	Passed
29	Others	Passed
30	User interaction security	Passed

3 Project Overview

3.1 Project Introduction

Audit version

Android

Version: 3.115.0

Sha256Sum: 5f1f912e97498b33a9ccd3059fe646f7faa6f3e50bf6a9ba8bf2fb0847e100c5

Fixed version

Android

Download Link: <https://pub-c0728063da7e456185643525139269c0.r2.dev/KuCoin-App-Release.apk>

Version: 3.119.0

Sha256Sum: 16d368ade79f0c015d74142883409b76ea6a5a88919211f30c2e2d8f94eb4825

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	App runtime environment detection Issue	App runtime environment detection	Suggestion	Acknowledged
N2	Code decompilation detection Issue	Code decompilation detection	Suggestion	Acknowledged
N3	App permissions detection Issue	App permissions detection	Suggestion	Acknowledged
N4	The app lacks mutual binding with certificates	Communication encryption security audit	Suggestion	Acknowledged
N5	Webview Default Enables Remote Debugging	WebView DOM security audit	Suggestion	Fixed
N6	Webview Lacks Whitelist Restriction for Access	WebView DOM security audit	Suggestion	Fixed
N7	Wallet Automatically Connects to Dapp	WebView DOM security audit	Suggestion	Acknowledged
N8	Dapp Signature Requests Can Be Sent Outside the WebView Scope	WebView DOM security audit	Low	Fixed
N9	Client-Based Authentication Issue	Client-Based Authentication Security audit	Suggestion	Acknowledged
N10	Signature security Issue	Signature security audit	Suggestion	Fixed

NO	Title	Category	Level	Status
N11	Secret key backup Issue	Secret key backup security audit	Suggestion	Acknowledged
N12	Screenshot/screen recording Issue	Screenshot/screen recording detection	Suggestion	Acknowledged
N13	Paste copy Issue	Paste copy detection	Suggestion	Acknowledged
N14	Keyboard keystroke cache Issue	Keyboard keystroke cache detection	Suggestion	Acknowledged
N15	Insecure entropy source Issue	Insecure entropy source audit	Suggestion	Acknowledged
N16	Background obfuscation Issue	Background obfuscation detection	Suggestion	Acknowledged
N17	Suspend evoke Issue	Suspend evoke security audit	Suggestion	Acknowledged
N18	AML anti-money laundering Issue	AML anti-money laundering security policy detection	Suggestion	Acknowledged
N19	ADB Log Leaks Sensitive Information Such as Mnemonic and Password	Others	High	Fixed
N20	User interaction Issue	User interaction security	Suggestion	Acknowledged

3.3 Vulnerability Summary

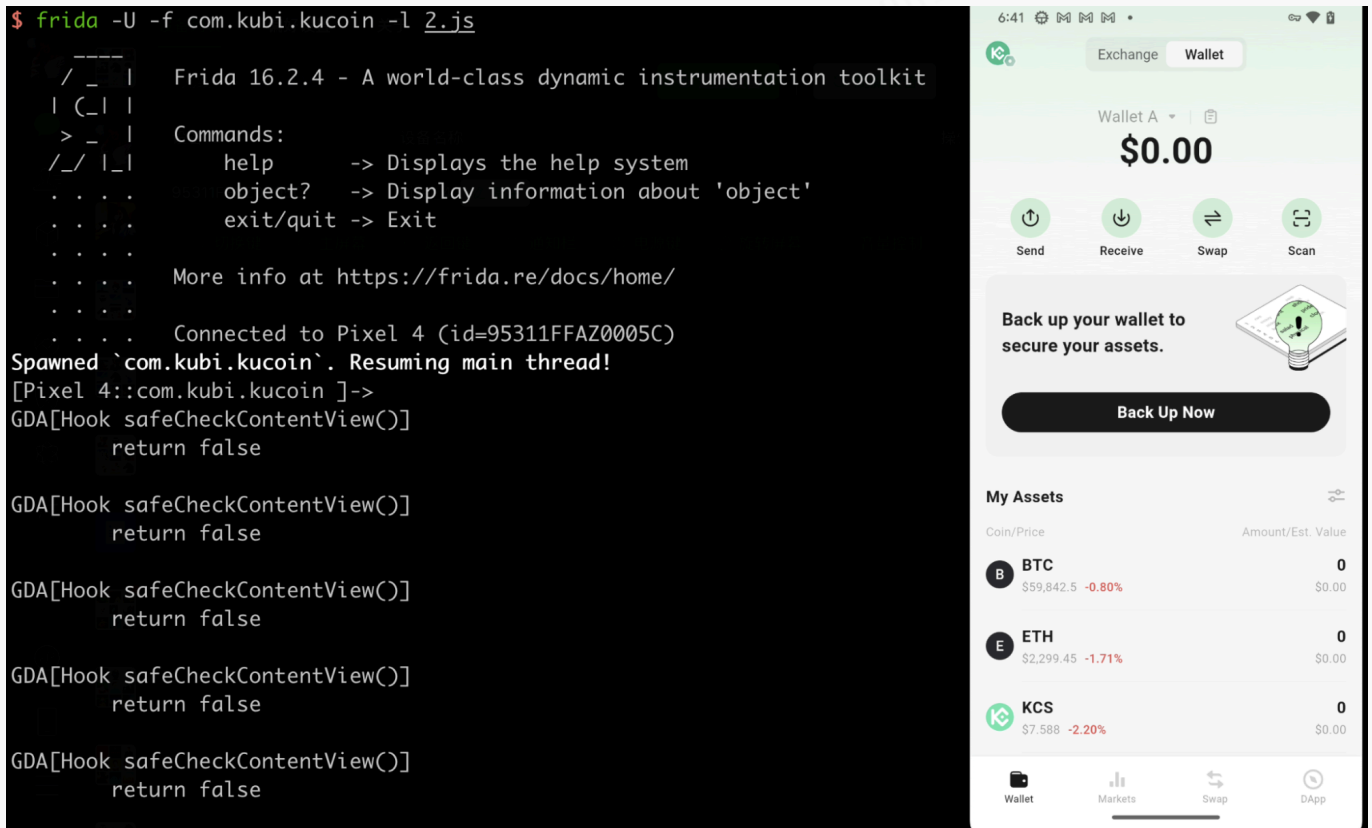
[N1] [Suggestion] App runtime environment detection Issue

Category: App runtime environment detection

Content

1. About root detection, the decompiled code includes checks for rooted devices, but these do not appear to be effective in practice.

2. About hook detection, no hooks detection were found, e.g., using Frida hooks to bypass root detection.



Solution

1. About root detection

When using an open-source solution, it's recommended to create your own detection rules.

Ensure these rules avoid using obvious keywords like "root."

Reference link: <https://mas.owasp.org/MASTG/Android/0x05j-Testing-Resiliency-Against-Reverse-Engineering/#testing-root-detection-mstg-resilience-1>

1. About hook detection

It's recommended to include detection for common hook frameworks like Frida and Xposed, and issue alerts accordingly.

Reference link:

<https://web.archive.org/web/20181227120751/http://www.vantagepoint.sg/blog/90-the-jiu-jitsu-of-detecting-frida>

Status

Acknowledged

[N2] [Suggestion] Code decompilation detection Issue

Category: Code decompilation detection**Content**

The KuCoin App claims to use the Enterprise version of Bangcle to secure the APK, but the decompiled code shows no signs of such protection or obfuscation, suggesting that the relevant security options might not have been selected during the fortification process.

Solution

It's recommended to ensure that fortification options are selected before deploying the APK in production.

Status

Acknowledged

[N3] [Suggestion] App permissions detection Issue**Category: App permissions detection****Content**

This app requires the following permissions:

```
Read image files from shared storage
permission: android.permission.READ_MEDIA_IMAGES
Read the contents of your shared storage
permission: android.permission.READ_EXTERNAL_STORAGE
Modify or delete the contents of your shared storage
permission: android.permission.WRITE_EXTERNAL_STORAGE
Read video files from shared storage
permission: android.permission.READ_MEDIA_VIDEO
Manage accounts
permission: android.permission.MANAGE_ACCOUNTS
```

When dealing with permissions, make sure users understand why the app needs them and only request what's absolutely necessary.

Solution

It's recommended to follow the principle of least privilege to minimize unnecessary permissions.

Status

Acknowledged

[N4] [Suggestion] The app lacks mutual binding with certificates**Category: Communication encryption security audit****Content**

Communication encryption performs certificate verification on the client-side and does not employ mutual authentication.

1. Communication encryption is carried out using the HTTPS protocol for transmission.
2. Communication encryption performs certificate verification on the client-side and does not employ mutual authentication.

Solution

1. In SSL two-way binding authentication, it's generally required to verify that the Certificate Authority (CA) is a trusted entity before checking the domain and public key information in the certificate.
2. SSL certificate two-way binding is generally more secure than one-way binding. One-way binding can be easily bypassed, allowing for the analysis of the app's network requests. While two-way binding poses greater difficulty to bypass, there are still techniques that may be employed.
3. Regardless of whether it's one-way or two-way binding, implementing certificate binding helps prevent man-in-the-middle attacks. However, during testing, one-way binding can be relatively easier to bypass for the purpose of analyzing app traffic.

Status

Acknowledged

[N5] [Suggestion] Webview Default Enables Remote Debugging**Category: WebView DOM security audit****Content**

By default, `AndroidWebViewController.enableDebugging` is set to true, enabling Remote WebView debugging and allowing developers to inspect and debug web content in Android apps using tools like Chrome DevTools.

- `kc_exchange_wallet-2024-08-19/lib/page/webview/webview_vm.dart#Line63-69`

```
if (controller.platform is AndroidWebViewController) {
  AndroidWebViewController.enableDebugging(true);
  (controller.platform as
AndroidWebViewController).setMediaPlaybackRequiresUserGesture(false);
}
```

- kc_wallet_dapp-2024-08-19/lib/src/dapp.dart#Line74-77

```
if (controller.platform is AndroidWebViewController) {
  AndroidWebViewController.enableDebugging(true);
  (controller.platform as
AndroidWebViewController).setMediaPlaybackRequiresUserGesture(false);
}
```

Solution

It is recommended to disable the ability for remote debugging tools to connect to WebViews within the app.

Status

Fixed

[N6] [Suggestion] Webview Lacks Whitelist Restriction for Access

Category: WebView DOM security audit

Content

Detecting if a WebView loads invalid scheme using a blacklist can be bypassed.

For example, by using the `data` or `ftp` protocols. Here's an example:

```
data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTWvc2NyaXB0Pg==
```

- kc_exchange_wallet-2024-08-19/lib/utls/dapp_helper.dart#Line20

```
const invalidScheme = ["javascript:", "file:///"];
```

- kc_exchange_wallet-2024-08-19/lib/utls/dapp_helper.dart#Line140-148

```
static String getFinalUrl(String? url) {
  if (url == null) {
    return 'https://www.google.com';
  }
}
```

```
}  
  
if (hasInvalidUrl(url)) {  
    return 'https://www.google.com';  
}
```

- [kc_exchange_wallet-2024-08-19/lib/utils/dapp_helper.dart#Line273-275](#)

```
static bool hasInvalidUrl(String url) {  
    return invalidScheme.any((scheme) => url.toLowerCase().contains(scheme));  
}
```

Solution

It is recommended to adopt a whitelist strategy that only allows URLs with `http:` and `https:` protocols, rejecting all others.

Status

Fixed

[N7] [Suggestion] Wallet Automatically Connects to Dapp

Category: WebView DOM security audit

Content

Accessing a Dapp wallet automatically connects without alerting the user or allowing them to decide whether to connect.

The current wallet design requires password verification for each signature, so even when automatically connecting to a Dapp and triggering a signature, password verification is needed. The project team has decided to maintain the status quo considering convenience.

Solution

It is recommended to let users decide whether to connect to a Dapp before linking their wallet.

Status

Acknowledged

[N8] [Low] Dapp Signature Requests Can Be Sent Outside the WebView Scope

Category: WebView DOM security audit**Content**

The wallet's signature requests on the DApp page are not scoped, allowing them to be received under other tabs like the Wallet, which poses a phishing risk.

Solution

It is recommended to restrict the scope of DApps by defaulting to disconnect and stop receiving requests when leaving the DApp page.

Status

Fixed

[N9] [Suggestion] Client-Based Authentication Issue**Category: Client-Based Authentication Security audit****Content**

The app lacks password verification when opened.

There's no password check when resuming from the background.

Solution

It is recommended to add password verification when opening the app and when resuming it from the background.

Status

Acknowledged

[N10] [Suggestion] Signature security Issue**Category: Signature security audit****Content**

- `kc_exchange_wallet-2024-08-19/lib/page/home_discover/dapp/view/dapp_sign_view.dart#Line582-602`

```
void onTapConfirm() {  
  if (widget.signType == SignMessageType.SMSignMessageType) {  
    KuToast.show(  
      message: WalletI10N.current.sign_type_not_support,  
    );  
  }  
}
```

```

    return;
  }
  WalletTrack.click(
    pageId: kPageIdDappTrans,
    blockId: 'SignPopupConfirm',
    params: {
      'result': 'success',
    },
  );
  AccountHelper.getPrivateKey(1).then((privateKey) {
    if (privateKey != null && privateKey.isNotEmpty) {
      processSign(context, privateKey);
    }
  });
}
}

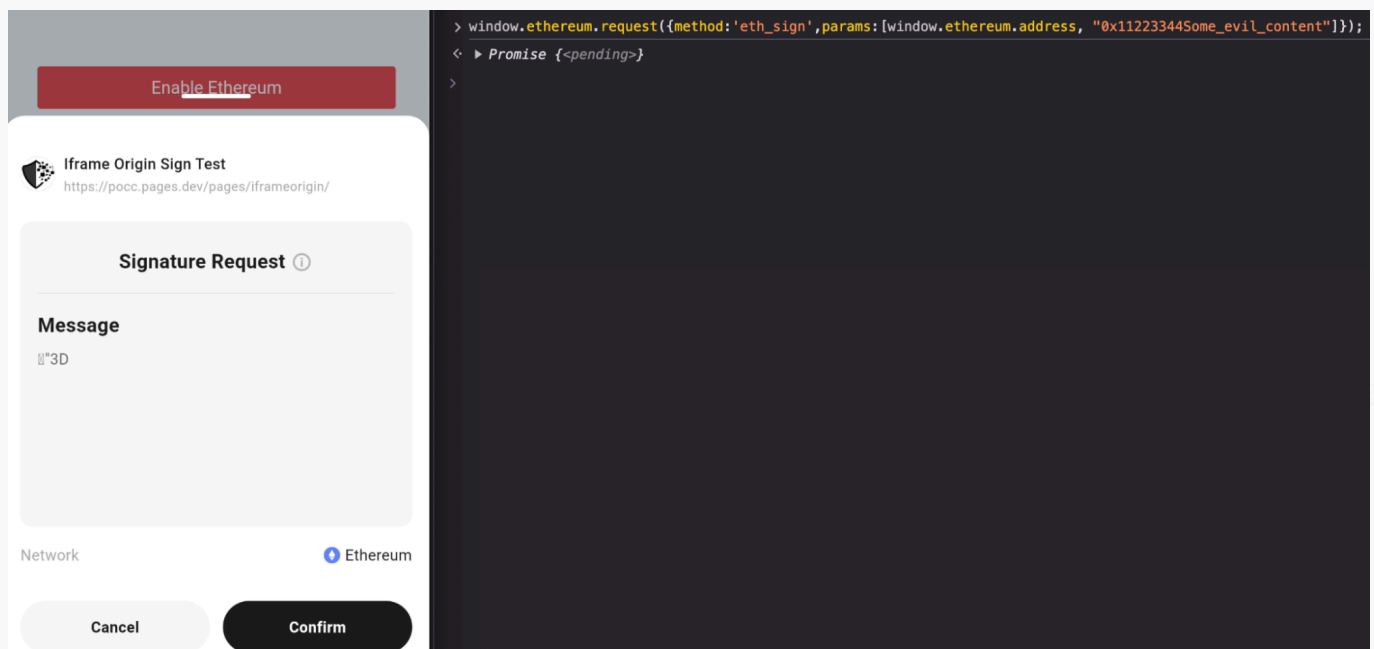
```

Using the signature data below will not trigger the "sign_type_not_support" alert.

```

window.ethereum.request({method: 'eth_sign', params: [window.ethereum.address,
"0x11223344Some_evil_content"]});

```

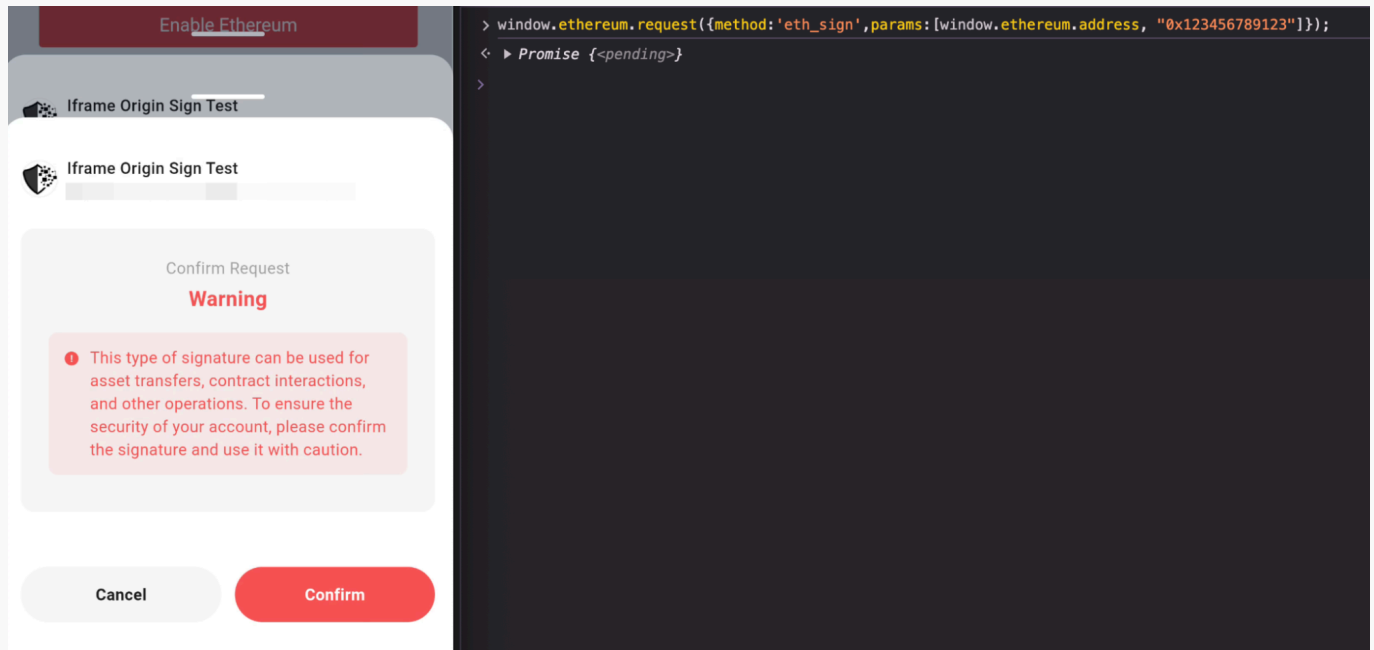


Using the signature data below will trigger the "sign_type_not_support" alert.

```

window.ethereum.request({method: 'eth_sign', params: [window.ethereum.address,
"0x123456789123"]});

```

Solution

It is recommended to promptly enhance compatibility checks to correctly identify blind signing methods like

`eth_sign`.

Status

Fixed

[N11] [Suggestion] Secret key backup Issue

Category: Secret key backup security audit

Content

There's a problem with only randomly verifying 3 words when backing up the mnemonic.

It doesn't provide enough validation.

Verify Seed Phrase

Select the correct words according to their numbers to verify that you've backed up the seed phrase.

No. 2

No. 3

No. 10

approve

limb

bracket

knee

tobacco

duck

again

border

double

day

wave

museum

Confirm

Solution

It is recommended to verify the full mnemonic phrase list when backing up.

Status

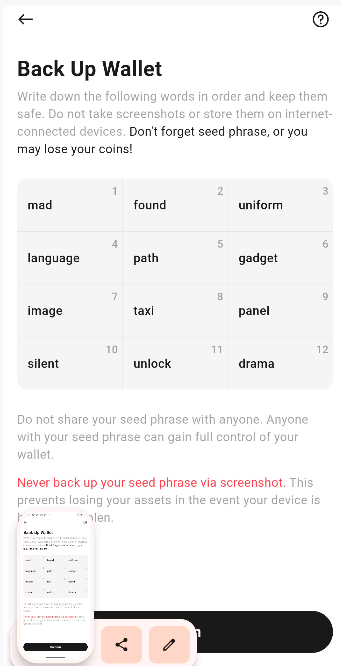
Acknowledged

[N12] [Suggestion] Screenshot/screen recording Issue

Category: Screenshot/screen recording detection

Content

The mnemonic backup page warns against screenshots and screen recordings, but it doesn't technically prevent them in the code.



Solution

To protect sensitive user information, screen capture/screen recording reminders should be added to all interfaces that involve such information.

Status

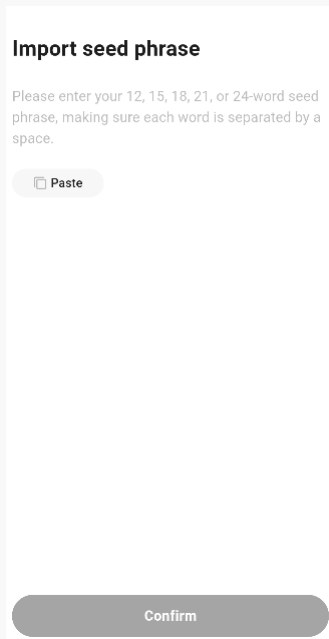
Acknowledged

[N13] [Suggestion] Paste copy Issue

Category: Paste copy detection

Content

When backing up, importing, or exporting, remind users to manually write down and type in mnemonic phrases rather than using copy-paste.



Many apps can access clipboard content, which might lead to mnemonic phrase leaks if copy-paste is used.

Solution

It is recommended to remind users to manually write down or type mnemonic phrases during backup, import, or export processes, avoiding the use of copy and paste.

Status

Acknowledged

[N14] [Suggestion] Keyboard keystroke cache Issue

Category: Keyboard keystroke cache detection

Content

The password and mnemonic phrases are entered using the mobile system's keyboard.

The app does not have its own secure keyboard.

Solution

It is recommended to add the functionality of a secure keyboard in the app. Using an in-app secure keyboard helps prevent third-party keyboards from accessing and sharing sensitive data.

Status

Acknowledged

[N15] [Suggestion] Insecure entropy source Issue

Category: Insecure entropy source audit

Content

`thread_rng()` in Rust's rand crate is not designed for cryptographic purposes. It is sufficiently secure for general random number generation tasks such as randomizing data, gaming, or simulations. However, it is not recommended for security-sensitive applications like generating encryption keys, passwords, or other scenarios requiring high-security measures.

- [app-security-sdk-master@934c494b449/src/features/rand.rs#Line23-27](#)

```
pub fn generate_secure_random_bytes(length: usize) -> Vec<u8>{
    let mut rng = thread_rng();
    let random_bytes: Vec<u8> = (0..length).map(|_| rng.gen::()).collect();
    random_bytes
}
```

- [app-security-sdk-master@934c494b449/src/features/rand.rs#Line6-20](#)

```
pub fn generate_secure_random_string(length: usize) -> String {
    const CHARSET: &[u8] = b"ABCDEFGHIJKLMNOPQRSTUVWXYZ\
                             abcdefghijklmnopqrstuvwxyz\
                             0123456789(*^%$#@!~";

    let mut rng = thread_rng();

    let password: String = (0..length)
        .map(|_| {
            let idx = rng.gen_range(0..CHARSET.len());
            CHARSET[idx] as char
        })
        .collect();

    return password;
}
```

Solution

It is recommended to use a cryptographically secure random number generator, such as OsRng, which directly utilizes the random sources provided by the operating system to ensure high-quality randomness.

Status

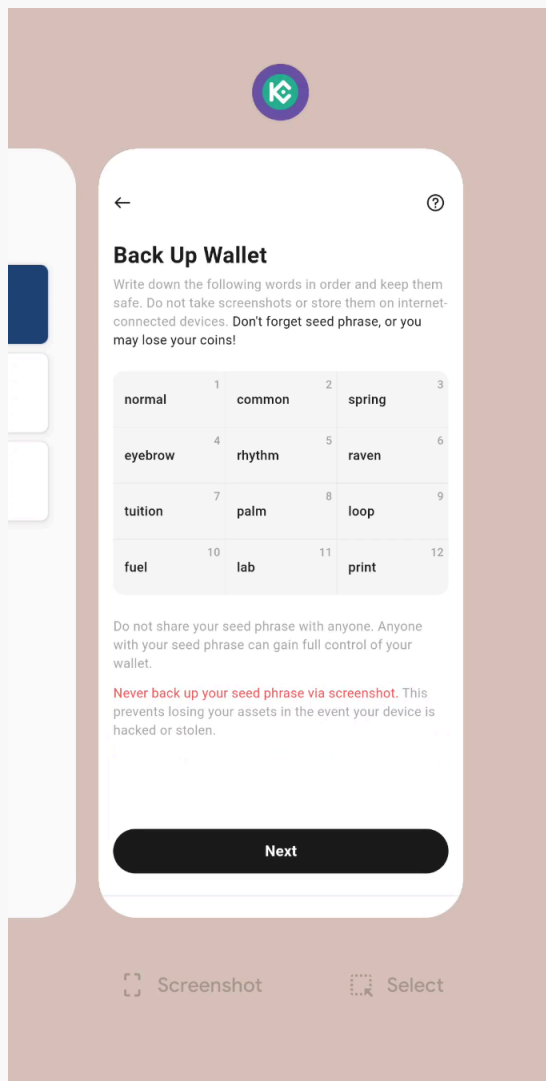
Acknowledged

[N16] [Suggestion] Background obfuscation Issue

Category: Background obfuscation detection

Content

The wallet app does not blur its screen when running in the background.



Solution

It is recommended to blur the wallet's screen when it is running in the background to prevent sensitive data from being exposed while switching apps, especially if the wallet is on a sensitive operation page.

Status

Acknowledged

[N17] [Suggestion] Suspend evoke Issue

Category: Suspend evoke security audit

Content

The wallet app lacks a timeout mechanism and does not require password verification when resuming from suspension.

Solution

It is recommended to implement a timeout mechanism in the wallet app and introduce password verification upon resuming from suspension.

Status

Acknowledged

[N18] [Suggestion] AML anti-money laundering Issue

Category: AML anti-money laundering security policy detection

Content

The app does not have access to the AML security policy and cannot synchronize malicious addresses to users in a timely manner.

Solution

It is recommended to access the AML security policy to remind users to avoid interacting with malicious addresses.

Status

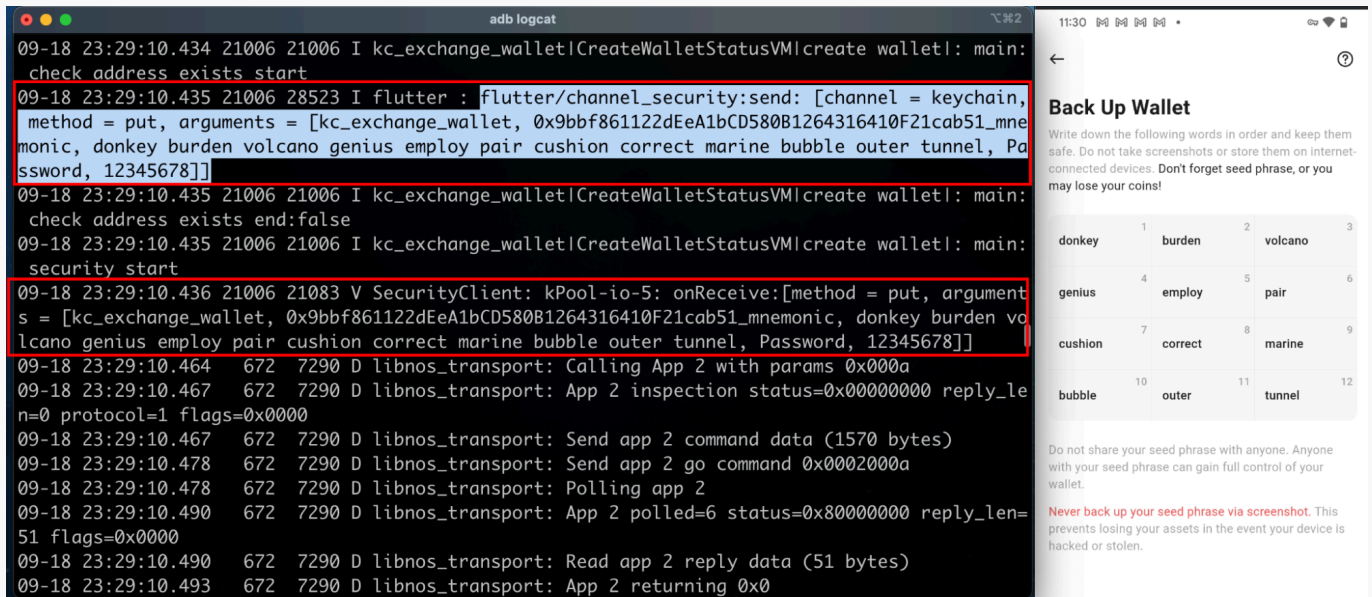
Acknowledged

[N19] [High] ADB Log Leaks Sensitive Information Such as Mnemonic and Password

Category: Others

Content

The app's log contains a lot of sensitive debug information, including mnemonic phrases and passwords.



Solution

It's recommended to block these details before going live in the production environment.

Status

Fixed

[N20] [Suggestion] User interaction Issue

Category: User interaction security

Content

Functionality	Support	Notes
WYSIWYS	•	Allow the use of eth_sign blind signing service.
AML	x	AML strategy is not supported.
Anti-phishing	x	Phishing detection warning is not supported.
Pre-execution	x	Pre-execution result display is not supported.
Contact whitelisting	•	The contact whitelisting is supported. Contact addresses can be added to the Address Book.
Password complexity requirements	x	There is no password complexity limit.

Tip: ✓ Full support, • Partial support, x No support

Solution

It is recommended to optimize relevant user interactions.

Status

Acknowledged

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002409040002	SlowMist Security Team	2024.08.22 - 2024.09.04	Passed

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 high risk, 2 low risk, 17 suggestion vulnerabilities. And 1 high risk and 2 low risks issue were fixed. All other findings were Acknowledged. We extend our gratitude for KuCoin wallet team recognition of SlowMist and hard work and support of relevant staff.

5 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.



Official Website
www.slowmist.com



E-mail
team@slowmist.com



Twitter
[@SlowMist_Team](https://twitter.com/SlowMist_Team)



Github
<https://github.com/slowmist>