



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 2022.07.15, the SlowMist security team received the team's security audit application for Sender Wallet iOS, 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/grey box lead, white box assists" 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.

Level	Description
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	Fixed
2	Code decompilation detection	Confirmed
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	Confirmed
8	WebKit security audit	Passed
9	App cache security audit	Passed
10	WebView DOM security audit	Passed

NO.	Audit Items	Result
11	SQLite storage security audit	Passed
12	Deeplinks security audit	Passed
13	Client-Based Authentication Security audit	Confirmed
14	Signature security audit	Passed
15	Deposit/Transfer security audit	Fixed
16	Transaction broadcast security audit	Passed
17	Mnemonic phrase/Private key generation security audit	Passed
18	Mnemonic phrase/Private key storage security audit	Fixed
19	Mnemonic phrase/Private key usage security audit	Fixed
20	Mnemonic phrase/Private key backup security audit	Passed
21	Mnemonic phrase/Private key destroy security audit	Passed
22	Screenshot/screen recording detection	Confirmed
23	Paste copy detection	Confirmed
24	Keyboard keystroke cache detection	Confirmed
25	Background obfuscation detection	Fixed
26	Suspend evoke security audit	Passed
27	AML anti-money laundering security policy detection	Confirmed
28	Others	Fixed

3 Project Overview

3.1 Project Introduction

Audit Version

<https://github.com/SenderWallet/sender-wallet-mobile/tree/slowmist-v0.0.1>

commit: 1e50dddb7167ca4184f09ca5a2e549fc9a12f4e7

IOS sender_mobile-v0.0.1.ipa(SHA256):

2612b01f5319598592d7c823e75ae297b4eb141e69fe32c8c321d6d6afa52081

Fixed Version

<https://github.com/SenderWallet/sender-wallet-mobile/tree/slowmist-v0.0.1>

commit: 1e50dddb7167ca4184f09ca5a2e549fc9a12f4e7

IOS sender_wallet_mobile-slowmist-fixed.ipa(SHA256):

70a3af77d8b6e6153285121a842497178be5c7194c4803d5c722ea4bce503fa7

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Runtime environment detection issues	App runtime environment detection	Low	Fixed
N2	Decompilation security issues	Code decompilation detection	Low	Confirmed
N3	Client Security Issues	Client-Based Authentication Security audit	Suggestion	Confirmed
N4	KeyStore lacks PBKDF2 protection	Mnemonic phrase/Private key storage security audit	Medium	Fixed

NO	Title	Category	Level	Status
N5	Missing screenshot/screen recording detection	Screenshot/screen recording detection	Suggestion	Confirmed
N6	Lack of security reminders	Paste copy detection	Suggestion	Confirmed
N7	Lack of secure keyboard	Keyboard keystroke cache detection	Suggestion	Confirmed
N8	Background obfuscation issue	Background obfuscation detection	Suggestion	Fixed
N9	Lack of AML security policy	AML anti-money laundering security policy detection	Suggestion	Confirmed
N10	Enhanced mnemonic verification	Business security audit	Suggestion	Confirmed
N11	signer is not cleared after expiration	Mnemonic phrase/Private key usage security audit	Medium	Fixed
N12	Strengthen reminder	Others	Suggestion	Fixed
N13	Missing status flag for transfer	Deposit/Transfer security audit	Low	Fixed
N14	Redundant code	Others	Suggestion	Fixed

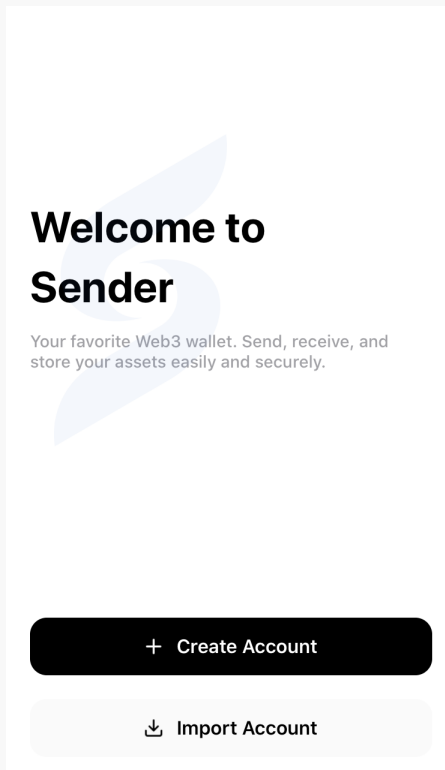
3.3 Vulnerability Summary

[N1] [Low] Runtime environment detection issues

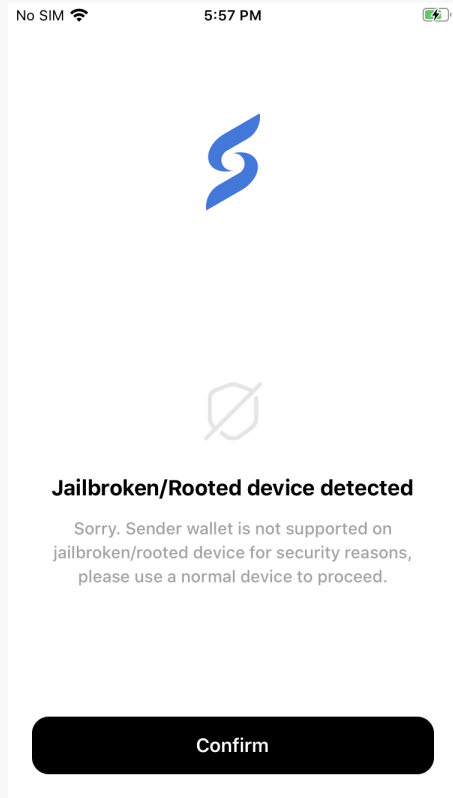
Category: App runtime environment detection

Content

When the jailbroken iPhone is installed for the first time, open the app to create or import an account. At this time, the device is not prompted to jailbreak.



After the user has created or imported an account, the reminder to perform the jailbreak detection is late.



Solution

The jailbreak detection sequence of wallet installation needs to be advanced in advance, and the jailbreak detection function has a higher priority than the account creation and import functions.

Status

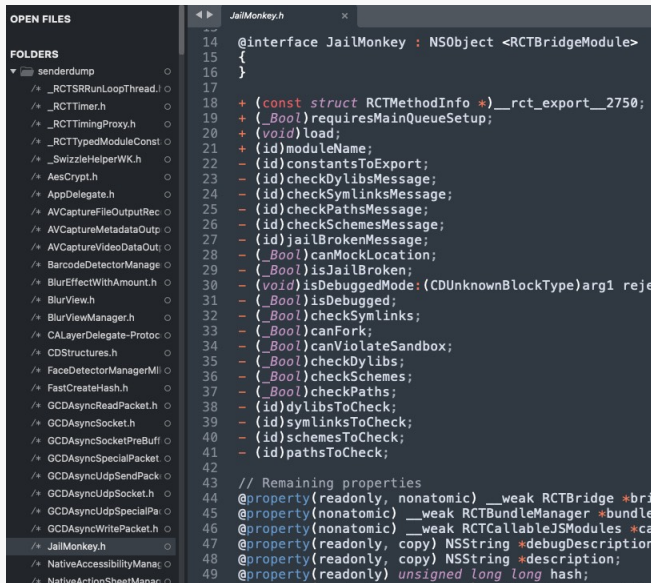
Fixed;

[N2] [Low] Decompilation security issues

Category: Code decompilation detection

Content

By dumping the `ipa` package, you can get the header file without code obfuscation of the header file.



```

14 @interface JailMonkey : NSObject <RCTBridgeModule>
15 {
16 }
17
18 + (const struct RCTMethodInfo *)__rct_export__2750;
19 + (_Bool)requiresMainQueueSetup;
20 + (void)load;
21 + (id)moduleName;
22 - (id)constantsToExport;
23 - (id)checkDylibsMessage;
24 - (id)checkSymlinksMessage;
25 - (id)checkPathsMessage;
26 - (id)checkSchemesMessage;
27 - (id)jailBrokenMessage;
28 - (_Bool)canMockLocation;
29 - (_Bool)isJailBroken;
30 - (void)isDebuggedMode:(CDUnknownBlockType)arg1 reject;
31 - (_Bool)isDebugged;
32 - (_Bool)checkSymlinks;
33 - (_Bool)canFork;
34 - (_Bool)canViolateSandbox;
35 - (_Bool)checkDylibs;
36 - (_Bool)checkSchemes;
37 - (_Bool)checkPaths;
38 - (id)dlibsToCheck;
39 - (id)symlinksToCheck;
40 - (id)schemesToCheck;
41 - (id)pathsToCheck;
42
43 // Remaining properties
44 @property(readonly, nonatomic) __weak RCTBridge *bridge;
45 @property(nonaatomic) __weak RCTBundleManager *bundleManager;
46 @property(nonaatomic) __weak RCTCallableJSMODULES *callables;
47 @property(readonly, copy) NSString *debugDescription;
48 @property(readonly, copy) NSString *description;
49 @property(readonly) unsigned long long hash;

```

Solution

It is recommended to obfuscate header files.

Status

Confirmed

[N3] [Suggestion] Client Security Issues

Category: Client-Based Authentication Security audit

Content

The background hangs when the app is restarted and the password authentication is not performed in time.

Solution

Sensitive pages are suspended and the screen is re-locked in the background without timely password authentication.

Status

Confirmed

[N4] [Medium] KeyStore lacks PBKDF2 protection

Category: Mnemonic phrase/Private key storage security audit

Content

The KeyStore encrypted storage uses the hashed password, but the password is hashed and stored without the pbkdf algorithm, which makes the password easy to be cracked by brute force.

- src/utils/crypto.js

```
import Aes from 'react-native-aes-crypto';

const CryptoJS = require('crypto-js');

export const generateHash = (password, salt) => {
  const hmac = CryptoJS.algo.HMAC.create(CryptoJS.algo.SHA512, salt);
  hmac.update(password);
  const hash = hmac.finalize();
  return hash.toString().substring(0, 64);
};

export const encrypt = (keyStore, key) => Aes.randomKey(16).then((iv) => {
  const text = JSON.stringify(keyStore);
  return (
    Aes.encrypt(text, key, iv, 'aes-256-cbc').then((cipher) => ({
      cipher,
      iv,
    }))
  );
});

export const decrypt = async (encryptedData, key) => {
  const message = await Aes.decrypt(encryptedData.cipher, key, encryptedData.iv, 'aes-256-cbc');
  return JSON.parse(message);
};
```

Solution

It is recommended to add pbkdf when encrypting storage to prevent brute force cracking.

Status

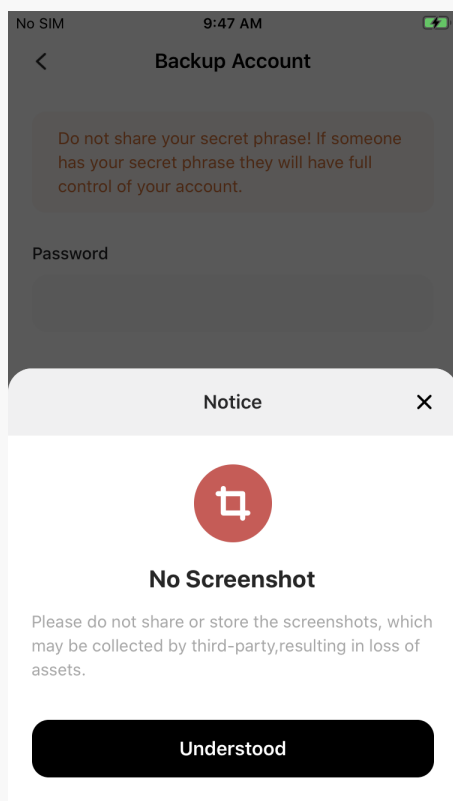
Fixed

[N5] [Suggestion] Missing screenshot/screen recording detection

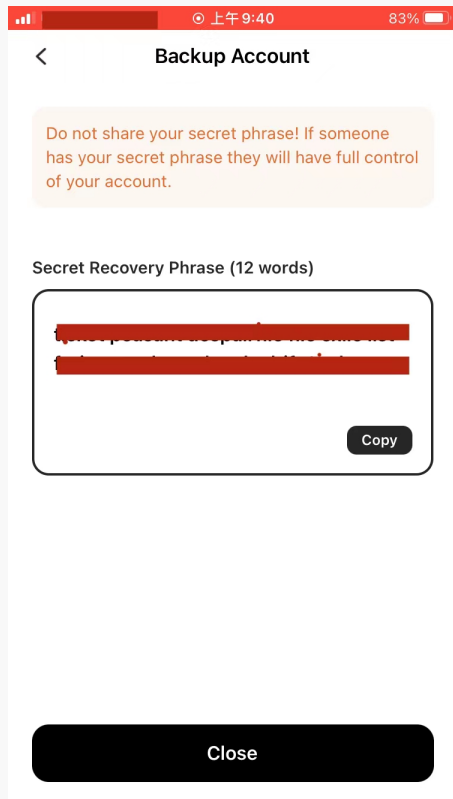
Category: Screenshot/screen recording detection

Content

When performing backup operations, there are reminders to take screenshots and screenshots.



But the app does not detect that the phone is taking screenshots or recordings.



Solution

It is recommended to monitor the screen capture and recording of the app by the system, and make a safety reminder.

Status

Confirmed

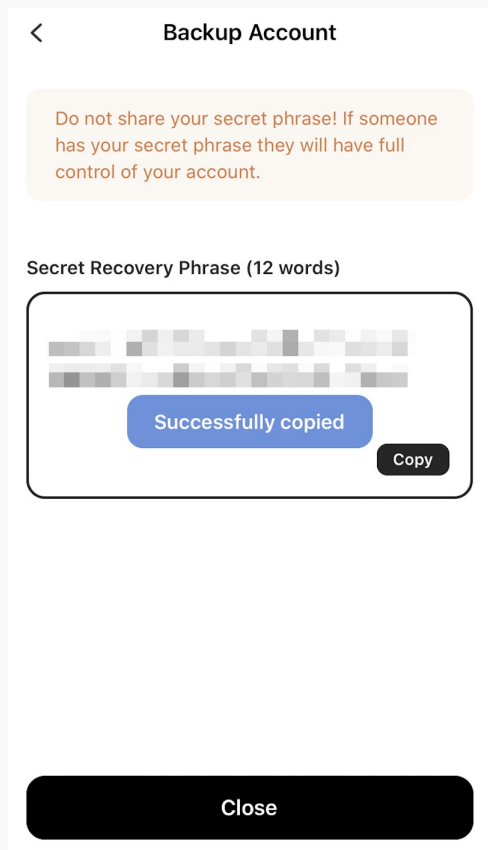
[N6] [Suggestion] Lack of security reminders

Category: Paste copy detection

Content

Copy the mnemonic with a reminder to copy.

However, the user is not reminded that it is best not to use the copy function, but to copy and so on.



Solution

It is recommended to remind users that it is best not to use the copy function, but to copy and so on.

Status

Confirmed

[N7] [Suggestion] Lack of secure keyboard

Category: Keyboard keystroke cache detection

Content

The app uses the system's own keyboard.

Solution

Use the keyboard that comes with the system to enter passwords, mnemonics, etc., and do not use a secure keyboard, which is easy to steal the keyboard cache.

Status

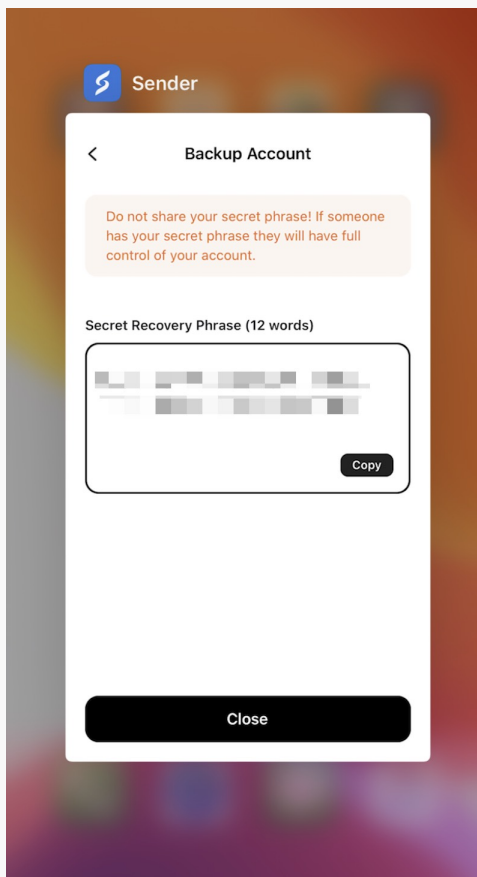
Confirmed

[N8] [Suggestion] Background obfuscation issue

Category: Background obfuscation detection

Content

App hangs without obfuscation.



Solution

It is recommended to monitor the app suspend operation to blur the background of the app.

Status

Fixed

[N9] [Suggestion] Lack of AML security policy

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

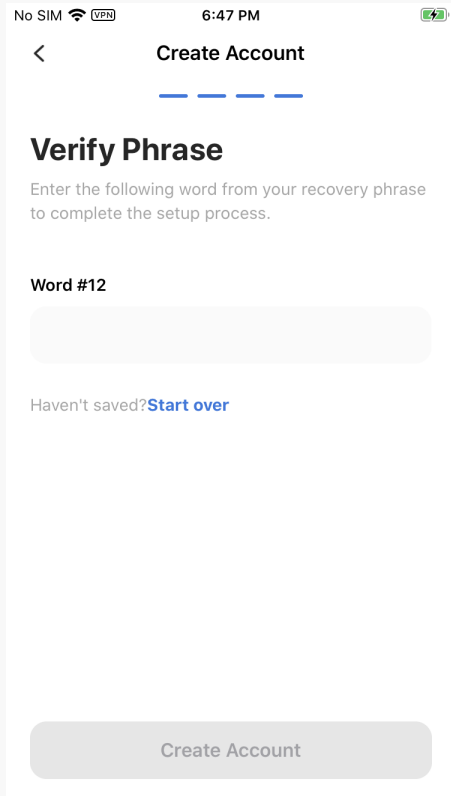
It is recommended to access the AML security policy to remind users to avoid interacting with malicious addresses.

Status

Confirmed

[N10] [Suggestion] Enhanced mnemonic verification**Category: Business security audit****Content**

When creating a wallet, the user is required to confirm whether the mnemonic phrase is backed up completely. The app only requires the user to verify 1 of the 12 mnemonic phrases, and this verification method needs to be strengthened. Because all mnemonics may not be fully backed up with.



Solution

It is recommended to scramble the 12 mnemonics and then let the user reorder the mnemonics, so as to guide the user to verify the correctness of each mnemonic.

Status

Confirmed

[N11] [Medium] signer is not cleared after expiration

Category: Mnemonic phrase/Private key usage security audit

Content

cleanPassword will be executed after the interval expires but this.signer is not assigned to null.

- src/screens/Home/Wallet/index.js line 192-209

```
useEffect(() => {
  const subscription = AppState.addListener('change', async (nextAppState) =>
  {
    appState.current = nextAppState;
```

```
const currentActiveTime = Date.now();
if (nextAppState === 'active') {
  const intervalTime = currentActiveTime - activeTime;
  if (intervalTime >= (LOCK_MINUTES * 60000) && keyStore) {
    const hashedPassword = await getPassword();
    if (hashedPassword) {
      cleanPassword();
      // to unlock page
      navigation.push('Unlock');
    }
  }
} else {
  dispatch(setActiveTime(currentActiveTime));
}
});
```

- src/core/near.js line 127-138

```
export class NearService {
  constructor({ config, accountId }) {
    this.viewAccount = getViewAccount({ config, accountId });
    this.signer = null;
    this.config = config;
    this.apiHelper = new ApiHelper({ helperUrl: config.helperUrl });
  }

  setSigner = async ({ mnemonic, accountId }) => {
    const signer = await getSigner({ mnemonic, accountId, config: this.config });
    this.signer = signer;
  };
}
```

Review records:

Added `cleanSigner` function.

- src/core/near.js#line293-330

```
cleanSigner = async () => {
  this.signer = null;
};
```

Code fix, `nearService.cleanSigner` will be called after each call to `nearService.setSigner` to clean up.

- `src/redux/sagas/near.js#308-311`

```
function* transferSaga(action) {
  const { asset, amount, receiver } = action;
  const appStore = yield select(getAppStore);
  const nearStore = yield select(getNearStore);
  const { currentAccount, currentRpc } = nearStore;
  const nearService = new NearService({ config: currentRpc });
  let hash = '';
  try {
    const { keyStore } = appStore;
    const password = yield call(getPassword);
    const { accountId } = currentAccount[currentRpc.network];

    const decryptedKeyStore = yield call(decrypt, keyStore, password);
    const { mnemonic } = decryptedKeyStore[accountId];

    const formatAmount = formatAssetBalance(amount, asset.decimals);
    yield call(nearService.setSigner, { mnemonic, accountId });
    const response = yield call(nearService.transfer, { contractId: asset?.accountId
|| '', amount: `${formatAmount}`, receiverId: receiver });
    yield call(nearService.cleanSigner);

    if (response.error) {
      hash = response.error?.transaction_outcome.id;
      throw new Error(getErrorMessage(response));
    }

    const params = {
      asset, amount, receiver, date: Date.now(), hash: response.transaction.hash,
    };

    RootNavigation.navigate('Home/Wallet/Send/Success', params);
  } catch (error) {
    const params = {
      asset, amount, receiver, date: Date.now(), hash, error: error.message,
    };
    yield call(nearService.cleanSigner);
    RootNavigation.navigate('Home/Wallet/Send/Failed', params);
  }
}
```

```
}  
}
```

Solution

It is recommended to also clear the value of this.signer when executing cleanPassword.

Status

Fixed

[N12] [Suggestion] Strengthen reminder**Category: Others****Content**

When importing the wallet, if the mnemonic is wrong, the app will not prompt the import error, but stay in the importing.

Solution

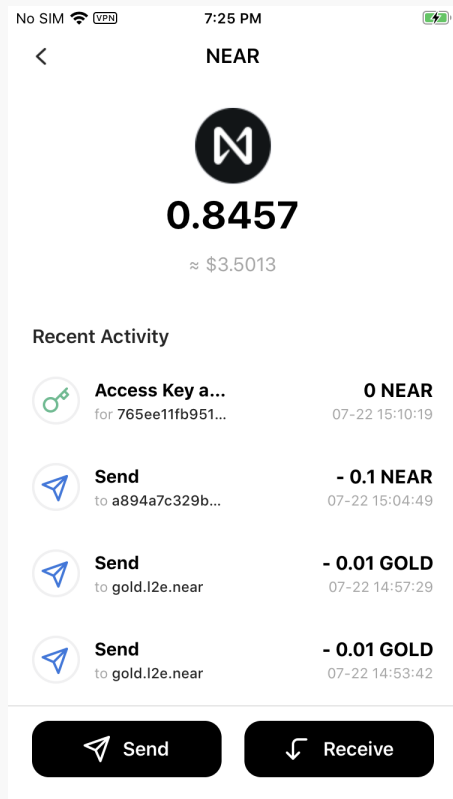
It is recommended to remind the user after the wallet import fails to avoid confusion for the user without an error prompt.

Status

Fixed

[N13] [Low] Missing status flag for transfer**Category: Deposit/Transfer security audit****Content**

Transfer failures are not flagged in the app, and incorrect transfer status may be used for scams. All transfers of GOLD tokens are failed.



Solution

It is recommended to flag the status of the transfer in the app.

Status

Fixed

[N14] [Suggestion] Redundant code

Category: Others

Content

There are useless commented code in the file and code that is not used in actual business.

- src/screens/Home/Settings/index.js line 83-88

```
const settings = [
  // {
  //   onPress: () => { navigation.navigate('Wallet/Manage'); },
  //   icon: require('../../../../assets/img/settings-faceid.png'),
  //   label: 'Face ID',
  //   rightComponent: <Image source={require('../../../../assets/img/arrow-
```

```
right.png')) />,  
  // },
```

src/screens/Home/Settings/index.js line 185-196

```
    { /* <Text style={[styles.fontSemiBold, styles.fontSize14,  
styles.lineHeight20, { color: '#262626', marginTop: scaleSize(32) }]}>Support</Text>  
    {  
      _.map(support, (item) => {  
        return <SettingItem  
          key={item.label}  
          onPress={item.onPress}  
          icon={item.icon}  
          label={item.label}  
          rightComponent={item.rightComponent}  
        />  
      })  
    } */ }
```

Solution

It is recommended to remove redundant commented code and useless code.

Status

Fixed

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002207260002	SlowMist Security Team	2022.07.15 - 2022.07.26	Low Risk

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 2 medium risk, 3 low risk, 9 suggestion vulnerabilities.

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>