



# Browser Extension Wallet Security Audit Report



# Table Of Contents

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

# 1 Executive Summary

On 2022.06.07, the SlowMist security team received the Sender Wallet team's security audit application for Sender Wallet, 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 DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi 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 browser extension wallet 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 browser extension wallets are manually analyzed to look for any potential issues.

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

- Transfer security
  - Signature security audit
  - Deposit/Transfer security audit
  - Transaction broadcast security audit
- Private key/Mnemonic phrase security
  - Private key/Mnemonic phrase generation security audit
  - Private key/Mnemonic phrase storage security audit
  - Private key/Mnemonic phrase usage security audit
  - Private Key/Mnemonic backup security audit
  - Private Key/Mnemonic destroy security audit
  - Random generator security audit
  - Cryptography security audit
- Web front-end security
  - Cross-Site Scripting security audit

- Third-party JS security audit
- HTTP response header security audit
- Communication security
  - Communication encryption security audit
  - Cross-domain transmission security audit
- Architecture and business logic security
  - Access control security audit
  - Wallet lock security audit
  - Business design security audit
  - Architecture design security audit
  - Denial of Service security audit

## 3 Project Overview

### 3.1 Project Introduction

#### Audit Version

<https://github.com/SenderWallet/sender-wallet-extension/tree/slowmist-new-ui-version>

commit: 83ad85892a7b18e8cccd08f27c9b286c674e0555

#### Fixed Version

<https://github.com/SenderWallet/sender-wallet-extension/tree/slowmist-new-ui-version>

commit: 26e4c7f5a51979fd03cf7f7189ce1dd7b2f64b52

### 3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Password is stored in local storage	Architecture design security audit	Low	Confirmed
N2	Authorized Apps page information display issue	Others	Suggestion	Confirmed
N3	The security issues with packages	Third-party JS security audit	Low	Fixed

### 3.3 Vulnerability Summary

#### [N1] [Low] Password is stored in local storage

**Category:** Architecture design security audit

##### Content

Passwords are stored in localStorage. In the unlocked case, The encrypted KeyStore can be decrypted with the password. But the password will be cleared after the wallet is locked.

- [src/pages/Background/handleExtensionRequest.js#L83-L100](#)

```

case APP_LOCK: {
  await updatePassword('');
  cleanInjectAccountId();
  chrome.alarms.clearAll();
  response = true;
  break;
}

case APP_UNLOCK: {
  const { password } = request;
  const hashedPassword = await generateKey(password);
  await getKeyStore(hashedPassword);
  await updatePassword(hashedPassword);
}

```

```
response = true;

updateAlarms();
break;
}
```

## Solution

It is recommended to store passwords in memory.

## Status

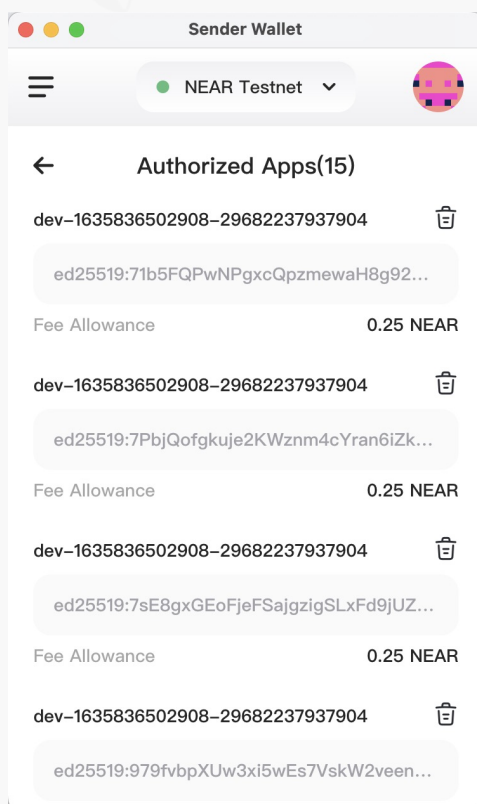
Confirmed; The project team response: manifest version 3 cannot persist password in the service worker.

## [N2] [Suggestion] Authorized Apps page information display issue

### Category: Others

### Content

Authorized Apps page does not display website URL.



### Solution

It is recommended to add the display of the URL.

### Status

Confirmed

## [N3] [Low] The security issues with packages

### Category: Third-party JS security audit

### Content

minimist@1.2.5 and json-schema@0.2.3 are used in the project code, and they have prototype pollution vulnerabilities,

The following are the dependencies of the package:

```
json-schema@0.2.3 -> jsprim@^1.2.2: -> http-signature@~1.2.0  
minimist@1.2.5 -> json5@1.0.1 -> loader-utils@1.4.0 -> react-hot-loader@4.13.0
```

### Solution

It is recommended to update the version of the package based on the following dependencies.

<https://snyk.io/test/npm/json5/1.0.1?tab=dependencies>

<https://snyk.io/test/npm/http-signature/1.2.0?tab=dependencies>

### Status

Fixed

## 4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206130002	SlowMist Security Team	2022.06.07 - 2022.06.13	Low Risk



Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 low risk, 1 suggestion vulnerabilities. And 1 low risk, 1 suggestion vulnerabilities were confirmed All other findings were fixed.

## 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](http://www.slowmist.com)



**E-mail**  
[team@slowmist.com](mailto:team@slowmist.com)



**Twitter**  
[@SlowMist\\_Team](https://twitter.com/SlowMist_Team)



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