Browser Extension Wallet Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.04.06, 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/e64edfee25c48226e443ee62905e92b136e38545

Fixed Version

https://github.com/SenderWallet/sender-wallet-extension/tree/1d9e804bd2a41978a13f1d8d65c43e7b0827a247

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Locked state can be bypassed	Wallet lock security audit	Low	Fixed
N2	Not using PBKDF to protect the KeyStore	Cryptography security audit	Medium	Fixed
N3	Password is stored in local storage	Architecture design security audit	Low	Confirmed
N4	Redundant configurations	Others	Suggestion	Fixed
N5	Authorized Apps page information display issue	Others	Suggestion	Confirmed

3.3 Vulnerability Summary

[N1] [Low] Locked state can be bypassed

Category: Wallet lock security audit

Content

Determine the status of the lock by getting the value of password in local storage.

- src/pages/Background/index.js#L761

```
const handleFromPageRequest = async (request, sender) => {
  try {
    const isLockup = !(await getPassword())

    if (isLockup && request.method !== 'disconnect' && request.method !== 'signOut')
{
      request = { ...request, method: 'toHomePage', requestMethod: request.method }
    }
    ......
```

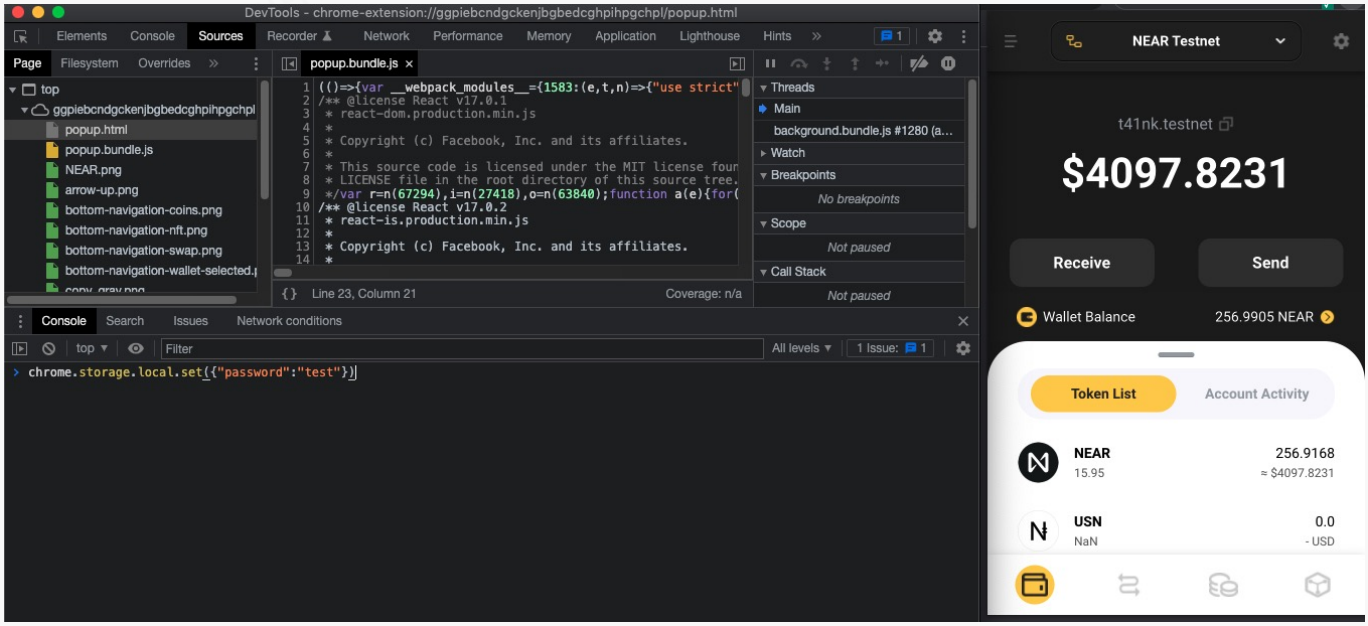
- src/pages/Background/index.js#L504-L505

```
      case APP_GET_LOCK_STATUS: {
        const password = await getPassword()
        response = !password
        break
      }
```

- src/pages/Background/index.js#L47

```
const getPassword = () => {
  return getData('password')
}
```

When the wallet is locked, an arbitrary value can be set for password to bypass the locked verification, but the private key cannot be decrypted, and only private data such as transactions can be viewed.



Solution

It is recommended to use the password to determine whether the KeyStore can be correctly unlocked to determine the unlocking status.

Status

Fixed

[N2] [Medium] Not using PBKDF to protect the KeyStore

Category: Cryptography security audit

Content

tweetnacl has not been maintained for a long time, and no implementation of PBKDF was found.

- src/core/crypto.js

```
import { secretbox, randomBytes } from 'tweetnacl'
// import sha256 from 'crypto-js/sha256'
import {
  decodeUTF8,
  encodeUTF8,
  encodeBase64,
  decodeBase64
} from 'tweetnacl-util'

const newNonce = () => randomBytes(secretbox.nonceLength)

export const generateKey = async (password) => {
  const hash = new TextEncoder().encode(password)
  const arrayBuffer = await crypto.subtle.digest('SHA-256', hash)
  return encodeBase64(new Uint8Array(arrayBuffer))
}

export const encrypt = (json, key) => {
  const keyUint8Array = decodeBase64(key)

  const nonce = newNonce()
  const messageUint8 = decodeUTF8(JSON.stringify(json))
  const box = secretbox(messageUint8, nonce, keyUint8Array)

  const fullMessage = new Uint8Array(nonce.length + box.length)
  fullMessage.set(nonce)
  fullMessage.set(box, nonce.length)

  const base64FullMessage = encodeBase64(fullMessage)
  return base64FullMessage
}

export const decrypt = (messageWithNonce, key) => {
```

```
  const keyUint8Array = decodeBase64(key)
  const messageWithNonceAsUint8Array = decodeBase64(messageWithNonce)
  const nonce = messageWithNonceAsUint8Array.slice(0, secretbox.nonceLength)
  const message = messageWithNonceAsUint8Array.slice(
    secretbox.nonceLength,
    messageWithNonce.length
  )

  const decrypted = secretbox.open(message, nonce, keyUint8Array)

  if (!decrypted) {
    throw new Error('Could not decrypt message')
  }

  const base64DecryptedMessage = encodeUTF8(decrypted)
  return JSON.parse(base64DecryptedMessage)
}
```

Solution

It is recommended to use encryption and decryption in browser-passworder to manage private keys and mnemonics.

reference: https://github.com/MetaMask/browser-passworder

Status

Fixed

[N3] [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/index.js#L509-L518

```
      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

[N4] [Suggestion] Redundant configurations

Category: Others

Content

There are redundant configurations in the manifest.json.

- src/manifest.json

```
{
  "manifest_version": 3,
  "version": "0.3.0",
  "name": "Sender Wallet",
  "options_page": "options.html",
  "background": { "service_worker": "background.bundle.js" },
  "action": {
    "default_popup": "popup.html",
    "default_icon": "icon-38.png"
  },
  "icons": {
    "128": "icon-128.png"
  },
```

```
  "content_scripts": [
    {
      "matches": ["http://*/*", "https://*/*", "<all_urls>"],
      "js": ["contentScript.bundle.js", "script.bundle.js"],
      "css": ["content.styles.css"],
      "run_at": "document_start",
      "all_frames": true
    }
  ],
  "devtools_page": "devtools.html",
  "web_accessible_resources": [
    {
      "resources": ["content.styles.css", "icon-128.png", "icon-38.png",
"script.bundle.js", "nearApiJs.bundle.js"],
      "matches": ["http://*/*", "https://*/*", "<all_urls>"]
    }
  ],
  "host_permissions": ["https://helper.mainnet.near.org/*",
"https://helper.testnet.near.org/*"],
  "permissions": [
    "storage",
    "alarms"
  ]
}
```

Solution

It is recommended to remove redundant configuration such as <all_urls> , devtools_page and options_page.

reference:

https://developer.chrome.com/docs/extensions/mv3/match_patterns/

Status

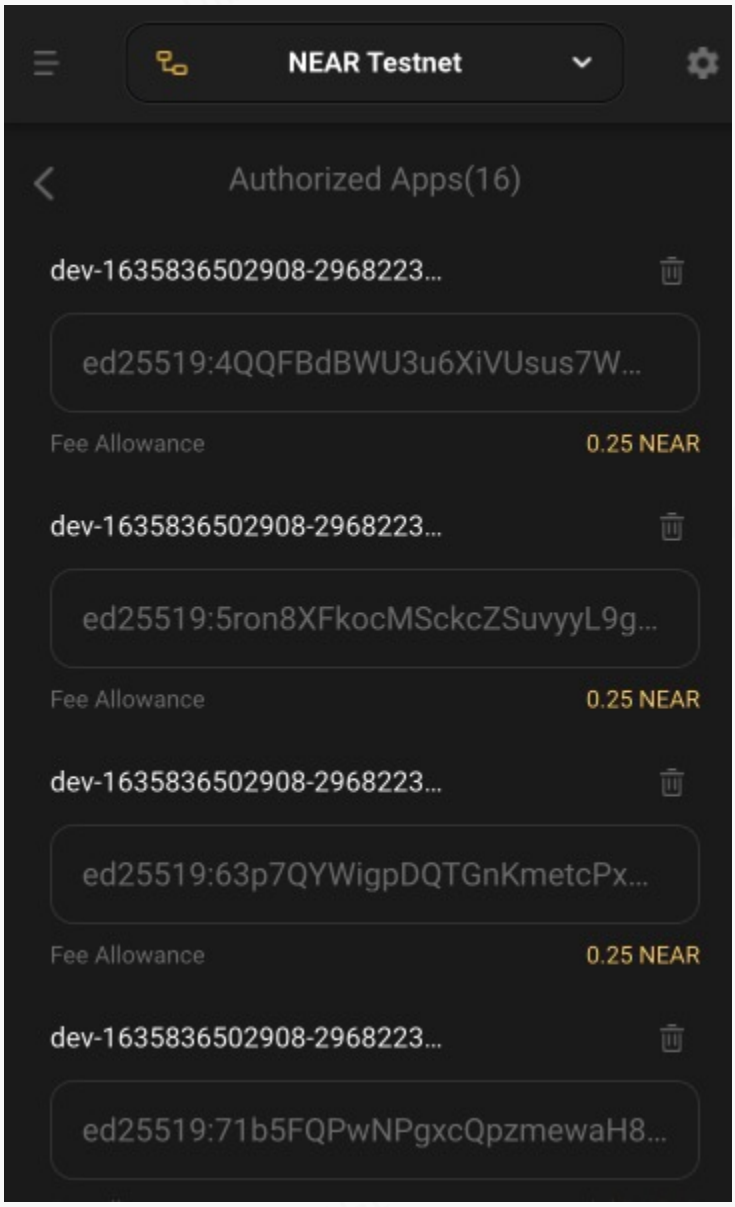
Fixed

[N5] [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

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002204120003	SlowMist Security Team	2022.04.06 - 2022.04.12	Passed

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 a medium risk, 2 low risk vulnerabilities, 2 suggestions. And a low risk, a suggestion were confirmed and 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

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist