Browser Extension Wallet Security Audit Report

Table of Contents

1 Executive Summary

On 2022.02.29, the AegisOne security team received the team's security audit application for Pitaka browser extension 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 AegisOne security team adopts the strategy of "black/gray 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:

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	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.
Suggestion	There are better practices for coding or architecture

2 Audit Methodology

The security audit process of AegisOne security team for browser extension wallet includes two steps:

The codes are scanned/tested for commonly known and more specific vulnerabilities using 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 phrase backup security audit
    - Private Key/Mnemonic phrase destroy security audit
- Web front-end security
    - Cross-Site scripting 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/tetrixtech/pitaka-wallet/releases/tag/Pitaka_v0.1.5

Fixed Version:

https://github.com/tetrixtech/pitaka-wallet/releases/tag/Pitaka_v0.1.6

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Signature source not reminded	Signature security audit	Low	Confirmed
N2	Design Optimization Recommendation	Others	Suggestion	Fixed

N3	Parse transactions can be bypassed	Business design security audit	Medium	Fixed
N4	Permission check is missing	Access control security audit	Low	Fixed
N5	"False Top-up" Vulnerability	Deposit/Transfer security audit	Low	Confirmed
N6	Code optimization	Wallet lock security audit	Suggestion	Fixed

3.3 Vulnerability Summary

[N1] [Low] Signature source not reminded

Category: Signature security audit

Content

When interacting with the DApp, Pitaka does not reveal the DApp domain origin of the request signature, which makes it easy for users to be confused.

Solution

It is recommended to display the signed domain origin when interacting with the DApp.

Status

Confirmed

[N2] [Suggestion] Design Optimization Recommendation

Category: Others

Content

If the DApp actively requests to connect to the Pitaka wallet, after the Pitaka wallet refuses, the DApp page continues to request the connection multiple times, and the wallet has no mechanism to prevent malicious connection requests.

Solution

It is recommended that the user should only be allowed to connect to the DApp from the wallet after the user rejects the DApp's automatic connection.

Status

Fixed; The project response: Changed net_version and eth_chainId to be called without connecting first.

[N3] [Medium] Parse transactions can be bypassed

Category: Business design security audit

Content

Since EVM's ABI will parse the input data when the input data is not long enough to be parsed, EVM will automatically pad this parameter with 0. This is how the classic short address attack works.

Pitaka wallet also has this kind of issue:

When the Approve function is executed normally, the input parameter length is correct, Pitaka will parse out the Approve transaction and display the specific amount.

By modifying the length of the amount parameter, the detection of this transaction type can be bypassed.

Solution

It is recommended to remind users if the data in the transaction has an abnormal length.

Status

Fixed.

[N4] [Low] Permission check is missing

Category: Access control security audit

Content

In the locked state, all functions related to the use of mnemonics and private keys need to verify the password. But popup.html#/history will still show the data of historical transactions.

- src/ui/views/MainRoute.tsx

Solution

It is recommended to add permission checks for each router's pages, which cannot be accessed when the wallet is locked.

Status

Fixed.

[N5] [Low] "False Top-Up" Vulnerability

Category: Deposit/Transfer security audit

Content

Due to a bug in Fantom's official node's parsing of internal transactions, internal transactions that fail are incorrectly parsed as successful.

Solution

It is recommended to judge the gas use of the internal transaction when parsing the internal transaction of Fantom. If the gas use is 0, it means that the internal transaction fails.

Status

Confirmed.

[N6] [Suggestion] Code optimization

Category: Wallet lock security audit

Content

this.password is repeatedly assigned.

- src/background/service/keyring/index.ts

```
async submitPassword(password: string): Promise<MemStoreState> {
await this.verifyPassword(password);
this.password = password;
try {
this.keyrings = await this.unlockKeyrings(password);
} catch {
//
} finally {
this.setUnlocked();
}
return this.fullUpdate();
}
```

- src/background/service/keyring/index.ts

```
async unlockKeyrings(password: string): Promise<any[]> {
const encryptedVault = this.store.getState().vault;
```

```
if (!encryptedVault) {
throw new Error(i18n.t('Cannot unlock without a previous vault'));
}
await this.clearKeyrings();
const vault = await this.encryptor.decrypt(password, encryptedVault);
this.password = password;
// TODO: FIXME
await Promise.all(Array.from(vault).map(this._restoreKeyring.bind(this)));
await this._updateMemStoreKeyrings();
return this.keyrings;
}
```

Solution

It is recommended to assign this.password after verifying the password and after the unlockKeyrings.

Status

Fixed.

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002203290012	SlowMist Security Team	2022.03.21 - 2022.03.31	Low Risk

Summary conclusion: The AegisOne security team uses a manual and AegisOne team's analysis tool to audit the project. During the audit, we found 1 medium risk, 3 low-risk, and 2 suggestion vulnerabilities. 3 low-risk vulnerabilities have been confirmed, all the other issues have been fixed.

5 Statement

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

For the facts that occurred or existed after the issuance, AegisOne 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 AegisOne by the information provider till the date of the insurance report (referred to as “provided information”). AegisOne 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 AEgisOne shall not be liable for any loss or adverse effect resulting therefrom. AegisOne only conducts the agreed security audit on the security situation of the project and issues this report. AegisOne is not responsible for the background and other conditions of the project.