Wallet Application Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.07.07, the SlowMist security team received the OneKey team's security audit application for OneKey SDK, 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 "white 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 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 party 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.

In black box testing and gray box testing, we use methods such as fuzz testing and script testing to test the robustness of the interface or the stability of the components by feeding random data or constructing data with a specific structure, and to mine some boundaries Abnormal performance of the system under conditions such as bugs or abnormal performance. In white box testing, we use methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to analyze the object definition and logic implementation of the code to ensure that the code has the key components of the key logic. Realize no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

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

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

NO.	Audit Items	Result
1	SAST	Passed
2	Business logic security audit	Some Risks

3 Project Overview

3.1 Project Introduction

This is the Javascript SDK for easily connecting OneKey hardware devices.

3.2 Coverage

Audit Version

https://github.com/OneKeyHQ/hardware-js-sdk

commit:97c47590300284a8be3b15c164a4a4850bcc0693

Core Scope:

- core
- hd-web-sdk
- hd-transport-webusb
- shared

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	isExtensionWhitelisted detection always returns True	Business logic security audit	Low	Confirmed
N2	Trust Degradation Subdomain Parsing Logic Flaw	Business logic security audit	Low	Confirmed
N3	Signature request smuggling	Business logic security audit	Low	Acknowledged

4 Findings

4.1 Vulnerability Summary

[N1] [Low] isExtensionWhitelisted detection always returns True

Category: Business logic security audit

Content

The `isExtensionWhitelisted` on the code does not take effect; any browser extension passing through this function will return true, and it is considered whitelisted. The business logic here needs to be clarified; whether whitelist detection is required, and if so, the whitelist list should be implemented.

Code location: /hardware-js-sdk/packages/hd-web-sdk/src/index.ts#L44

```
// easy to test and then open
// @ts-expect-error
export const isExtensionWhitelisted = (origin: string) => true;
// whitelistExtension.find(item => item === origin);
```

Solution

It is recommended to fully implement the functionality of the `isExtensionWhitelisted` function, set up the browser extension whitelist properly, rather than returning True for the judgment of any browser extension.

Status

Confirmed

[N2] [Low] Trust Degradation Subdomain Parsing Logic Flaw

Category: Business logic security audit

Content

The getHost function extracts the domain part of a URL using a regular expression and splits the domain into segments, retaining only the last two parts (e.g., sub.domain.com becomes domain.com). This logic implicitly trusts all subdomains as valid, leading to potential trust degradation issues.

Code location: /hardware-js-sdk/packages/hd-web-sdk/src/utils/urlUtils.ts#L9-L19

```
export const getHost = (url: unknown) => {
  if (typeof url !== 'string') return;
  const [, , uri] = url.match(/^(https?):\/\/([^:/]+)?/i) ?? [];
  if (uri) {
    const parts = uri.split('.');

    return parts.length > 2
      ? // slice subdomain
        parts.slice(parts.length - 2, parts.length).join('.')
      : uri;
  }
};
```

Code location: /hardware-js-sdk/packages/hd-web-sdk/src/index.ts#L39-L40

```
export const isOriginWhitelisted = (origin: string) => {
  const host = getHost(origin);

  return whitelist.find(item => item.origin === origin || item.origin === host);
};
```

The whitelist includes sources such as the following, which contain the file protocol, localhost, onekeytest.com, and other test sources, which may not be secure.

Code location: /hardware-js-sdk/packages/core/dist/index.js

```
const whitelist = [
    { origin: 'file://' },
    { origin: '1key.so' },
    { origin: 'onekey.so' },
    { origin: 'onekeycn.com' },
    { origin: 'onekeytest.com' },
    { origin: 'localhost' },
];
```

Solution

It is recommended to strictly constrain that the origin must be consistent with the whitelist.

Status

Confirmed

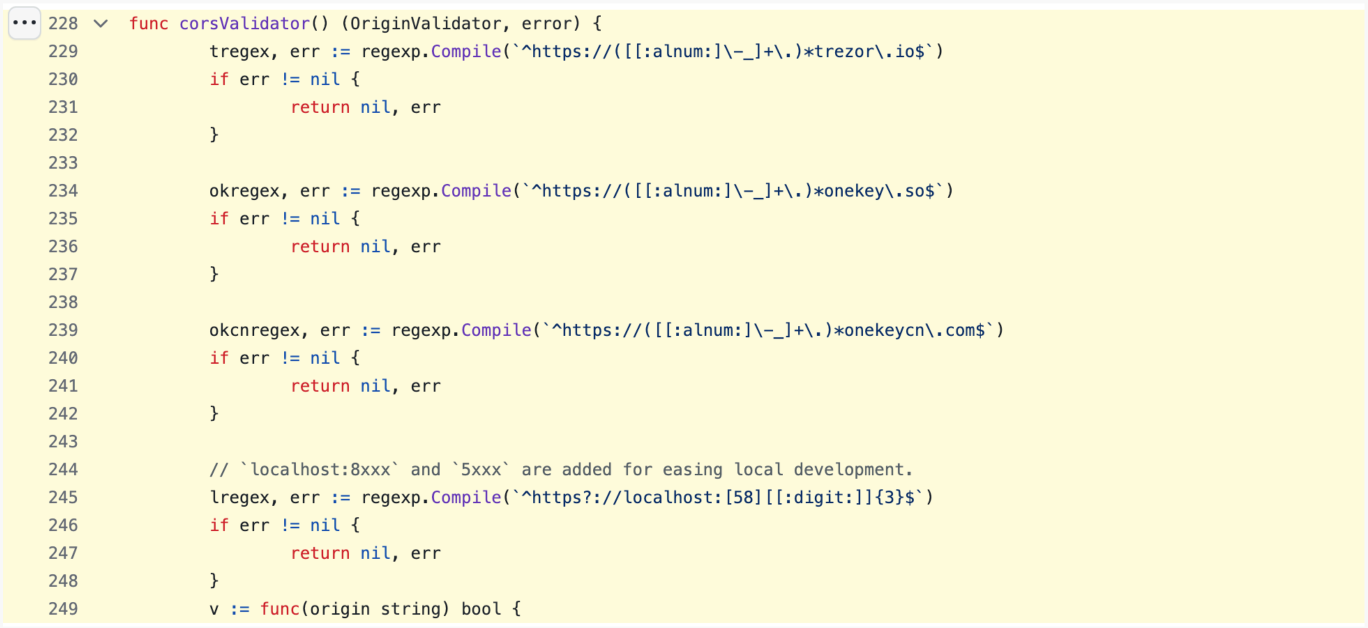
[N3] [Low] Signature request smuggling

Category: Business logic security audit

Content

The signature of a trusted dapp request can be replaced by the signature of a malicious dapp request. In specific, the hardware wallet displays the signature content of the trusted dapp, then the malicious dapp will close the trusted request after signing and reopen the new signature content. In this scenario, if the user does not observe that the request has been replaced, they may think that the signature data is from a trusted dapp. In the case of consecutive requests, the signature request from the trusted dapp may not even appear. OneKey Bridge is used to forward requests from jssdk.onekey.so. OneKey Bridge uses CORS to restrict communication to the following domains on http://localhost:21320/. However, jssdk.onekey.so allows any dapp to access and use it. jssdk.onekey.so does not check or pass the origin of dapp messages. Therefore, the hardware wallet cannot display the source of the signature request. The hardware wallet also allows unauthorized dapps to communicate with it, which leads to this issue.

Code location: https://github.com/OneKeyHQ/onekey-bridge/blob/onekey/server/api/api.go#L228



```go
func corsValidator() (OriginValidator, error) {
	tregex, err := regexp.Compile(`^https://([[:alnum:]\-_]+\.)*trezor\.io$`)
	if err != nil {
		return nil, err
	}

	okregex, err := regexp.Compile(`^https://([[:alnum:]\-_]+\.)*onekey\.so$`)
	if err != nil {
		return nil, err
	}

	okcnregex, err := regexp.Compile(`^https://([[:alnum:]\-_]+\.)*onekeycn\.com$`)
	if err != nil {
		return nil, err
	}

	// `localhost:8xxx` and `5xxx` are added for easing local development.
	lregex, err := regexp.Compile(`^https?://localhost:[58][[:digit:]]{3}$`)
	if err != nil {
		return nil, err
	}
	v := func(origin string) bool {
```

Solution

It is recommended to get the origin of the post message in jssdk.onekey.so and then pass it to the bridge. The bridge can pass it to the hardware wallet for authentication. Only dapps that have been authenticated can communicate with the hardware wallet.

Status

Acknowledged; After communication and feedback, the project party replied that they will close support for OneKey Bridge in the future, and currently, interaction with web3 extension wallets is only supported through the WebUSB method.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002507200001	SlowMist Security Team	2025.07.07 - 2025.07.20	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 3 low risk vulnerabilities.

6 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.