Browser Extension Wallet Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.12.02, the SlowMist security team received the Rabby team's security audit application for Rabby 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 SlowMist security team adopts the strategy of "black-box and grey-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 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.
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
- Secret key security
    - Secret key generation security audit
    - Secret key storage security audit
    - Secret key usage security audit
    - Secret key backup security audit
    - Secret key destruction 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/RabbyHub/Rabby

commit: 4e900e5944a671e99a135eea417bdfdb93072d99

Fixed Version

https://github.com/RabbyHub/Rabby

commit: d5a907385fe1815ba6e313f5d6630f2bc4827980

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Lack of Runtime Protections Mechanism	Others	Suggestion	Acknowledged
N2	Incorrect Configuration Information	Others	Suggestion	Fixed
N3	Matomo service may be unavailable	Denial of Service security audit	Suggestion	Acknowledged
N4	Lack of Risk Detection for Malicious Contract	User interaction security	Suggestion	Fixed
N5	Risk Detection Bypass	User interaction security	Low	Fixed

NO	Title	Category	Level	Status
N6	Poor Risk Detection of signTypedData Signature	User interaction security	Low	Fixed
N7	Detection of Contract Creation & Selfdestruct Phishing	User interaction security	Suggestion	Acknowledged

3.3 Vulnerability Summary

[N1] [Suggestion] Lack of Runtime Protections Mechanism

Category: Others

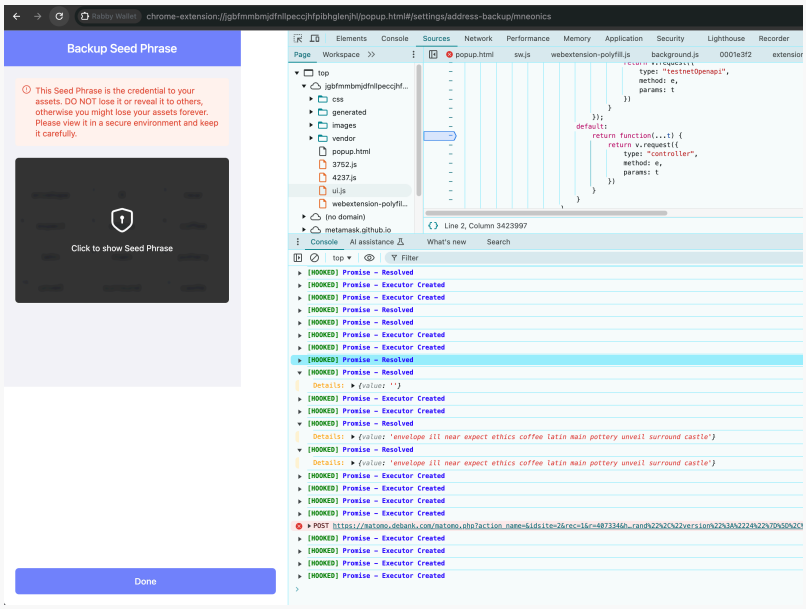
Content

Due to the lack of runtime protection mechanisms in the Rabby Extension Wallet, third-party dependencies or attackers can steal sensitive data by hooking native objects. For example, popups usually don't store data such as private keys and mnemonic phrases. However, when backing up or exporting the wallet, it's necessary to send the mnemonic phrases or private keys from Service Workers (background) to the popup for display, using the "controller" communication type. In this scenario, the mnemonic phrases or private keys can be captured by hooking the native types of `Promise`.

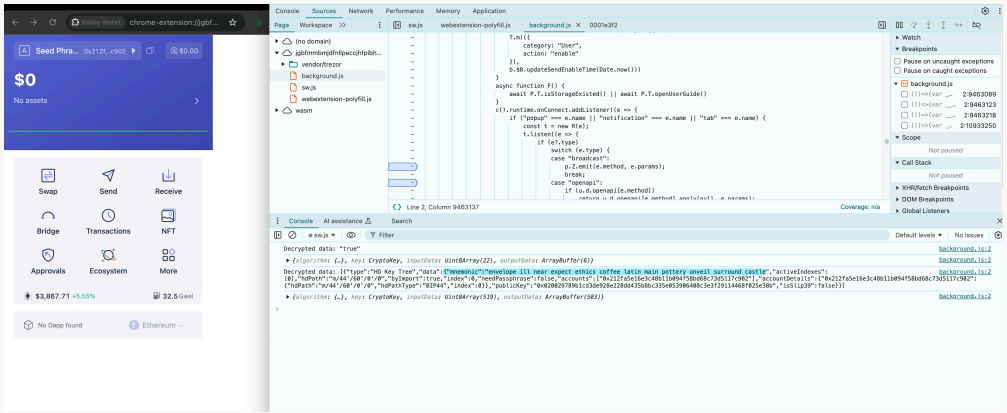
Code location: Rabby/src/ui/app.tsx#L57-L105

```
const wallet = new Proxy(
  {},
  {
    get(obj, key) {
      switch (key) {
        case 'openapi':
          return new Proxy(
            {},
            {
              get(obj, key) {
                return function (...params: any) {
                  return portMessageChannel.request({
                    type: 'openapi',
                    method: key,
                    params,
                  });
```

```
                };
              },
            }
          );
          break;
        case 'testnetOpenapi':
          return new Proxy(
            {},
            {
              get(obj, key) {
                return function (...params: any) {
                  return portMessageChannel.request({
                    type: 'testnetOpenapi',
                    method: key,
                    params,
                  });
                };
              },
            }
          );
          break;
        default:
          return function (...params: any) {
            return portMessageChannel.request({
              type: 'controller',
              method: key,
              params,
            });
          };
      }
    },
  }
) as WalletControllerType;
```



The design of the Rabby wallet is such that when unlocking the wallet, the mnemonic phrase is obtained by decryption in Service Workers (background). It uses the decrypt function in `@metamask/browser-passworder/dist/index.d.ts`, and this function invokes the method of `crypto.subtle.decrypt` for decryption. Therefore, it's possible to hook the decryption method of `crypto.subtle.decrypt` to directly obtain the plaintext mnemonic phrase from Service Workers (background) when unlocking the wallet.



Solution

It is recommended to use LavaMoat to freeze objects, preventing the modification of global variables through hooks in popups or Service Workers (background) to read wallet data.

Reference: https://github.com/LavaMoat/LavaMoat

Status

Acknowledged

[N2] [Suggestion] Incorrect Configuration Information

Category: Others

Content

In the `manifest.json` configuration file of the Rabby Extension Wallet, `web_accessible_resources` uses a non-existent resource named `user-media-permission.html`. This file has not been found in the project files.

And "vendor/matomo.js" and "/vendor/matomo.client.js" do not need to be injected into the web page, so they also belong to unnecessary configurations.

Code location: Rabby/src/manifest/mv3/manifest.json

```
"web_accessible_resources": [
    {
      "resources": [
        "user-media-permission.html",
        "pageProvider.js",
        "vendor/matomo.js",
        "/vendor/matomo.client.js"
      ],
      "matches": [
        "<all_urls>"
      ]
    }
```

Solution

It is recommended to delete unnecessary configuration information `user-media-permission.html`, `vendor/matomo.js` and `/vendor/matomo.client.js`.

Status

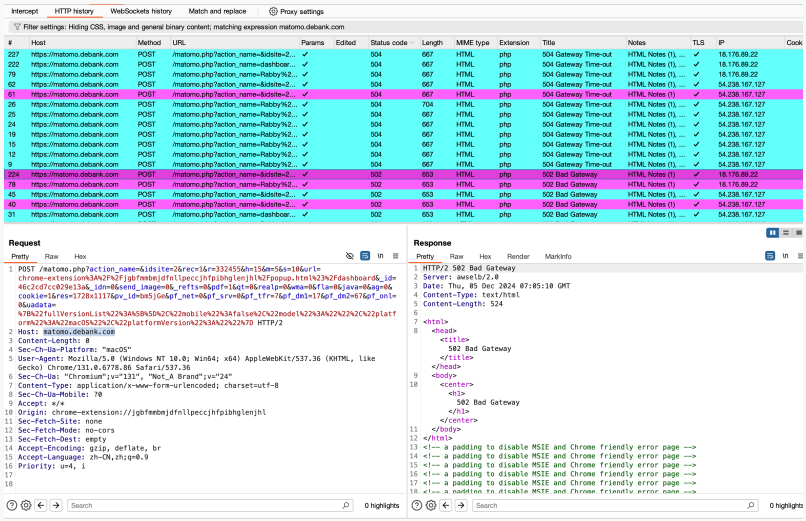
Fixed; This issue has been fixed in this commit: d5a907385fe1815ba6e313f5d6630f2bc4827980.

[N3] [Suggestion] Matomo service may be unavailable

Category: Denial of Service security audit

Content

The Rabby Extension Wallet uses "https://matomo.debank.com/matomo.php" to track users' operational behaviors and capture the operation logs of clients. However, errors 502 and 504 often occur on this interface, which may lead to the unavailability of related functions.



Solution

It is recommended to check the logs on the server side of "https://matomo.debank.com/matomo.php" so as to solve the problems of response status 502 and 504.

Status

Acknowledged

[N4] [Suggestion] Lack of Risk Detection for Malicious Contract

Category: User interaction security

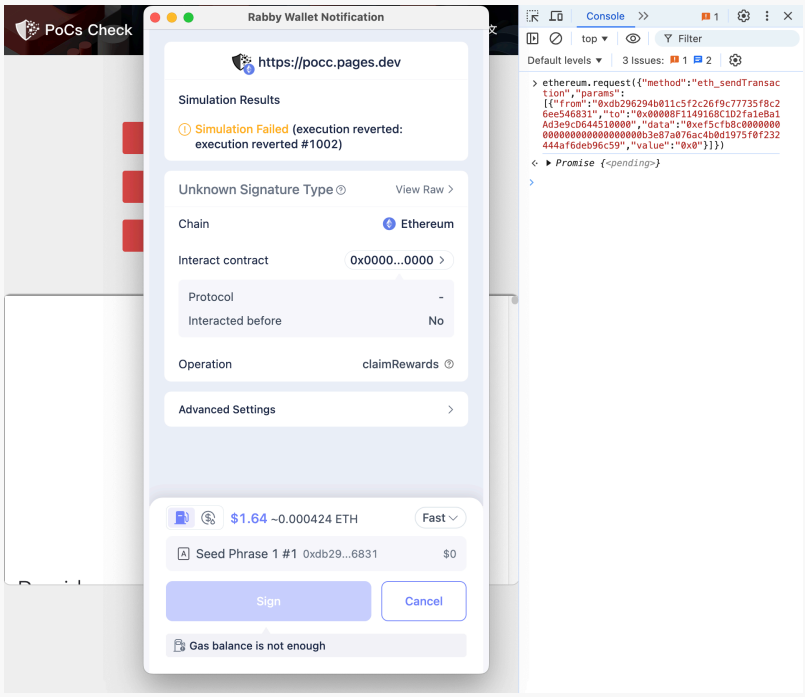
Content

Using the following Proof of Concept (PoC), the Rabby Extension Wallet fails to detect malicious contracts. Malicious contracts deceive users by using contract addresses in the format of "0x000...0000". Since most extension wallets do not display the complete contract address information, attackers take advantage of this situation to construct malicious contract addresses, deceive users into interacting with them, and thus steal users' assets.

The malicious contract "0x00008F1149168C1D2fa1eBa1Ad3e9cD644510000" has already been marked as "Phish / Hack", but the Rabby Extension Wallet does not give any risk warnings.

PoC:

```
ethereum.request({"method":"eth_sendTransaction","params":
[{"from":"0xdb296294b011c5f2c26f9c77735f8c26ee546831","to":"0x00008F1149168C1D2fa1eBa
1Ad3e9cD644510000","data":"0xef5cfb8c000000000000000000000000b3e87a076ac4b0d1975f0f2
32444af6deb96c59","value":"0x0"}]})
```



Solution

It is recommended to enhance the display of the format of addresses like "0x000...0000". Meanwhile, risk checks should be carried out on the contract addresses with which users interact, so as to help users identify the contract addresses that pose risks.

Status

Fixed

[N5] [Low] Risk Detection Bypass

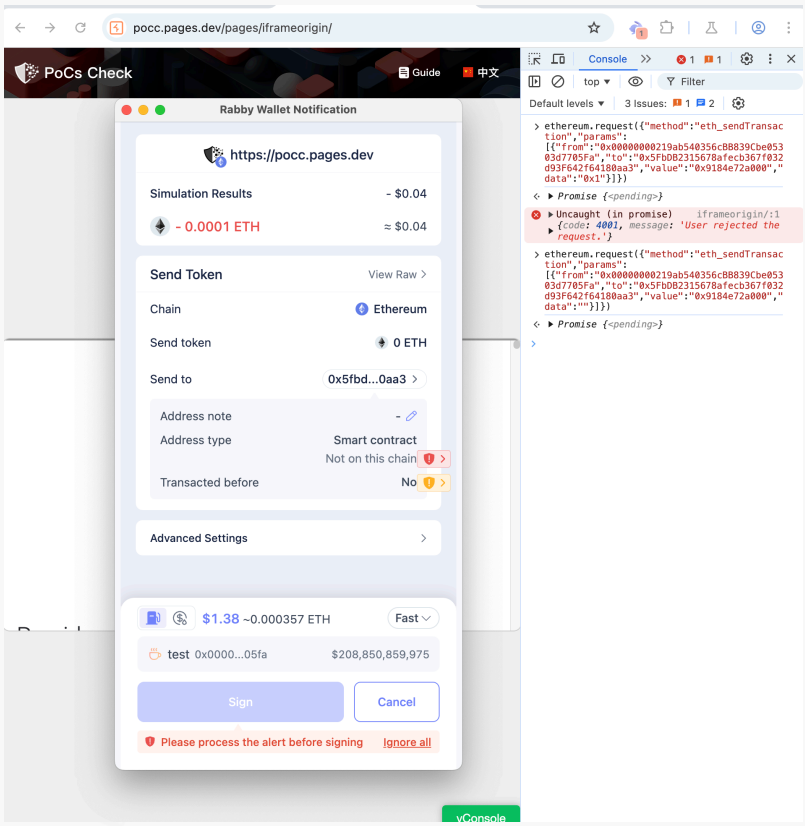
Category: User interaction security

Content

The Rabby Extension Wallet will conduct risk detection on the target wallet addresses with which users interact. The address `0x5FbDB2315678afecb367f032d93F642f64180aa3` has already been marked as `Phish / Hack`. There will be a risk warning if you interact with it directly. Please check the Source Request.

Source Request:

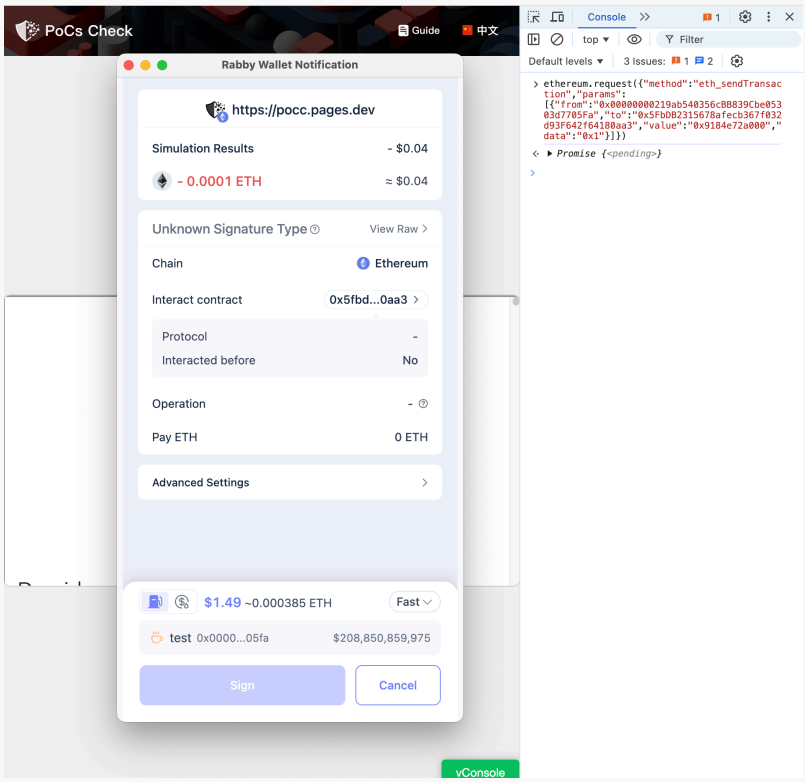
```
ethereum.request({"method":"eth_sendTransaction","params":
[{"from":"0x00000000219ab540356cBB839Cbe05303d7705Fa","to":"0x5FbDB2315678afecb367f03
2d93F642f64180aa3","value":"0x9184e72a000","data":""}]})
```



However, if any content is added to the "data" of the request, the Rabby Extension Wallet will not give any risk warnings. Since `0x5FbDB2315678afecb367f032d93F642f64180aa3` is an Externally Owned Account (EOA) address, the content of the "data" does not affect the transfer of ETH.

Bypass PoC:

```
ethereum.request({"method":"eth_sendTransaction","params":
[{"from":"0x00000000219ab540356cBB839Cbe05303d7705Fa","to":"0x5FbDB2315678afecb367f03
2d93F642f64180aa3","value":"0x9184e72a000","data":"0x1"}]})
```



Solution

It is recommended to conduct detections on the target addresses with which users interact. If the target addresses are malicious, risk warnings should be given regardless of what the data is.

Status

Fixed

[N6] [Low] Poor Risk Detection of signTypedData Signature

Category: User interaction security

Content

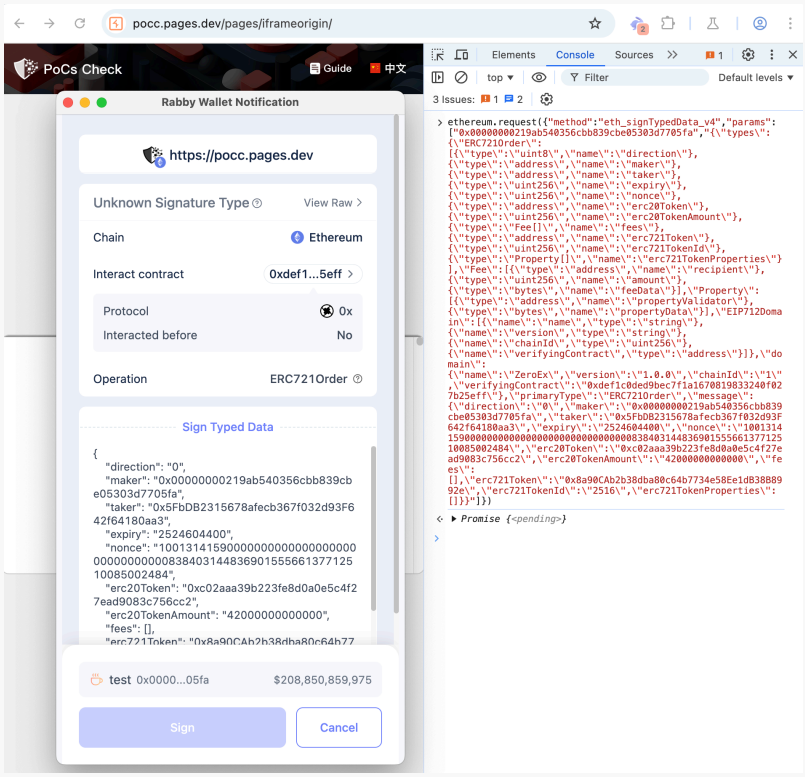
The address `0x5FbDB2315678afecb367f032d93F642f64180aa3` has already been marked as `Phish / Hack`.

However, in the signature type of `eth_signTypedData_v4`, when the content of the signature is to interact with the 0x protocol and the "taker" is a malicious address, the Rabby Extension Wallet will not identify the risk at this time.

PoC:

```
ethereum.request({"method":"eth_signTypedData_v4","params":
["0x00000000219ab540356cbb839cbe05303d7705fa","{\"types\":{\"ERC721Order\":
[{\"type\":\"uint8\",\"name\":\"direction\"},
{\"type\":\"address\",\"name\":\"maker\"},{\"type\":\"address\",\"name\":\"taker\"},
{\"type\":\"uint256\",\"name\":\"expiry\"},{\"type\":\"uint256\",\"name\":\"nonce\"},
{\"type\":\"address\",\"name\":\"erc20Token\"},
```

```
{\"type\":\"uint256\",\"name\":\"erc20TokenAmount\"},
{\"type\":\"Fee[]\",\"name\":\"fees\"},
{\"type\":\"address\",\"name\":\"erc721Token\"},
{\"type\":\"uint256\",\"name\":\"erc721TokenId\"},
{\"type\":\"Property[]\",\"name\":\"erc721TokenProperties\"}],\"Fee\":
[{\"type\":\"address\",\"name\":\"recipient\"},
{\"type\":\"uint256\",\"name\":\"amount\"},
{\"type\":\"bytes\",\"name\":\"feeData\"}],\"Property\":
[{\"type\":\"address\",\"name\":\"propertyValidator\"},
{\"type\":\"bytes\",\"name\":\"propertyData\"}],\"EIP712Domain\":
[{\"name\":\"name\",\"type\":\"string\"},{\"name\":\"version\",\"type\":\"string\"},
{\"name\":\"chainId\",\"type\":\"uint256\"},
{\"name\":\"verifyingContract\",\"type\":\"address\"}]},\"domain\":
{\"name\":\"ZeroEx\",\"version\":\"1.0.0\",\"chainId\":\"1\",\"verifyingContract\":\"0
xdef1c0ded9bec7f1a1670819833240f027b25eff\"},\"primaryType\":\"ERC721Order\",\"message
\":
{\"direction\":\"0\",\"maker\":\"0x00000000219ab540356cbb839cbe05303d7705fa\",\"taker\
":\"0x5FbDB2315678afecb367f032d93F642f64180aa3\",\"expiry\":\"2524604400\",\"nonce\":\
"100131415900000000000000000000000000000838403144836901555661377125100850024 84\",\"er
c20Token\":\"0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2\",\"erc20TokenAmount\":\"42000
000000000\",\"fees\":
[],\"erc721Token\":\"0x8a90CAb2b38dba80c64b7734e58Ee1dB38B8992e\",\"erc721TokenId\":\"
2516\",\"erc721TokenProperties\":[]}}"]})
```

Solution

It is recommended to strengthen the risk detection on the fields related to addresses in the signature data like signTypedData, so as to cover the risk detection of the above-mentioned issues.

Status

Fixed

[N7] [Suggestion] Detection of Contract Creation & Selfdestruct Phishing

Category: User interaction security

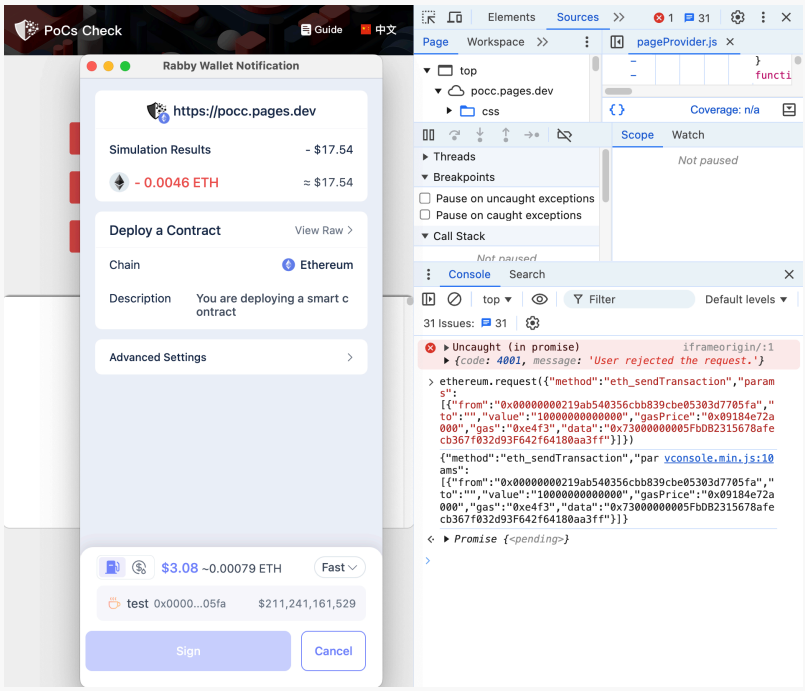
Content

The address `0x5FbDB2315678afecb367f032d93F642f64180aa3` has already been marked as `Phish / Hack`. However, by using `73` and `ff`, it's possible to destroy the contract when constructing it. After the contract is destroyed, ETH will be transferred to the designated address. Attackers will use this method to deceive users, making them think that they are creating a contract, while in fact, they are using the transaction of creating a contract to carry out the theft of coins.

The Rabby Extension Wallet didn't identify this type of phishing technique.

PoC:

```
ethereum.request({"method":"eth_sendTransaction","params":
[{"from":"0x00000000219ab540356cbb839cbe05303d7705fa","to":"","value":"10000000000000
","gasPrice":"0x09184e72a000","gas":"0xe4f3","data":"0x73000000005FbDB2315678afecb367
f032d93F642f64180aa3ff"}]})
```



Solution

It is recommended to add this kind of phishing techniques with relatively obvious characteristics and conduct detections on the already marked addresses so as to inform users of the existing risks. Detect phishing techniques that involve contract creation and self-destruct.

Status

Acknowledged

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002412060004	SlowMist Security Team	2024.12.02 - 2024.12.06	Passed

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 low risk vulnerabilities, 5 suggestions.

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