Wallet Application Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.09.07, the SlowMist security team received the Rabby team's security audit application for Rabby Wallet Desktop, 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 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 wallet application 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 wallet application is manually analyzed to look for any potential issues.

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

NO.	Audit Items	Result
1	App runtime environment detection	Passed
2	Code decompilation detection	Passed
3	App permissions detection	Passed
4	File storage security audit	Passed
5	Communication encryption security audit	Passed
6	Interface security audit	Passed
7	Business security audit	Passed
8	WebKit security audit	Passed
9	App cache security audit	Passed
10	WebView DOM security audit	Passed
11	SQLite storage security audit	Passed
12	Deeplinks security audit	Passed
13	Client-Based Authentication Security audit	Passed
14	Signature security audit	Passed
15	Deposit/Transfer security audit	Passed
16	Transaction broadcast security audit	Passed

NO.	Audit Items	Result
17	Secret key generation security audit	Passed
18	Secret key storage security audit	Passed
19	Secret key usage security audit	Passed
20	Secret key backup security audit	Passed
21	Secret key destruction security audit	Passed
22	Screenshot/screen recording detection	Passed
23	Paste copy detection	Passed
24	Keyboard keystroke cache detection	Passed
25	Insecure entropy source audit	Passed
26	Background obfuscation detection	Passed
27	Suspend evoke security audit	Passed
28	AML anti-money laundering security policy detection	Passed
29	Others	Confirmed
30	User interaction security	Passed

3 Project Overview

3.1 Project Introduction

Audit Version

Version: 0.33.0

Source: https://github.com/RabbyHub/RabbyDesktop/tree/v0.33.0-prod

commit: 586447a46bcd0abab6356076e369357050c97796

macOS

Silicon: https://download.rabby.io/wallet-desktop/darwin-arm64/rabby-wallet-desktop-installer-arm64-0.33.0.dmg

shasum256: b464a9bd3e5efbbb4d0c8ad87b81b11608dc82d2a1326266d60eb131fa3c0b09

Intel: https://download.rabby.io/wallet-desktop/darwin-x64/rabby-wallet-desktop-installer-x64-0.33.0.dmg

shasum256: f2db80ff027a5590f1f4dc0dd592d33c866b2fa30bc095480c49e53206f2d621

Windows

https://download.rabby.io/wallet-desktop/win32-x64/rabby-wallet-desktop-installer-x64-0.33.0.exe

shasum256:209cbc259ff89b673a32db53288c221d5cbf8fd96a65953b382354ff375564d6

Fixed Version

Version 0.34.0

Source:

https://github.com/RabbyHub/RabbyDesktop/pull/546

commit: dae23cb2dd249b76e17fbcaefb6f6f6af157d1f8

https://github.com/RabbyHub/RabbyDesktop/pull/547

commit: 14d01267229f2f828734080dc63b23115bb898fa

https://github.com/RabbyHub/RabbyDesktop/pull/548

commit: 312573445088036981ea06b49d0f6365a5176fa4

macOS

Silicon: https://download.rabby.io/versioned/rabby-wallet-desktop-reg/darwin-arm64/0.34.0-20230922_033857/rabby-wallet-desktop-installer-arm64-0.34.0-20230922_033857.dmg

shasum256: s0c02505e8bcdaa987713463c46162413a32abd261736dfc16aef9cb2abd7f7ba

Intel: https://download.rabby.io/versioned/rabby-wallet-desktop-reg/darwin-x64/0.34.0-20230922_033232/rabby-wallet-desktop-installer-x64-0.34.0-20230922_033232.dmg

shasum256: 34c79b3a0605b684a8423461ace0c79e4989b673c2e85963bfac54a5c930c729

Windows

https://download.rabby.io/versioned/rabby-wallet-desktop-reg/win32-x64/0.34.0-20230922_031656/rabby-wallet-

desktop-installer-x64-0.34.0-20230922_031656.exe

shasum256: d86594bd49fec4f304935bdf8a45205213e6a40ce11eff9fcd7e709b1d92290a

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Arbitrary File Read Issue	Others	Medium	Fixed
N2	Missing validation in shell.openExternal() leads to remote code execution	Others	Critical	Fixed
N3	Unverify the sender of all IPC messages.	Others	Suggestion	Fixed
N4	Windows version lack file integrity verification	Others	Low	Confirmed

3.3 Vulnerability Summary

Category: Others

Content

Rabby Wallet Desktop allows importing local DApp pages. When checking local files, it does not verify if the local file is a symbolic link file, which can lead to arbitrary file read issues.

- src/main/utils/dapps.ts#154-229

```
export async function detectLocalDapp(
  localDappPath: ICheckedOutDappURL | string,
  opts: {
    existedDapps: IDapp[];
  }
): Promise<IDappsDetectResult<DETECT_ERR_CODES>> {
  const checkedOutDappInfo =
    typeof localDappPath === 'string'
      ? checkoutDappURL(localDappPath)
```

```
    : localDappPath;
const inputOrigin = checkedOutDappInfo.dappOrigin;

const absPath =
  process.platform === 'win32'
    ? unPrefix(checkedOutDappInfo.localFSPath, '/')
    : checkedOutDappInfo.localFSPath;

if (!fs.existsSync(absPath)) {
  return {
    data: null,
    error: {
      type: DETECT_ERR_CODES.INACCESSIBLE,
      message: `The path doesn't exist.`,
    },
  };
}
if (!fs.statSync(absPath).isDirectory()) {
  return {
    data: null,
    error: {
      type: DETECT_ERR_CODES.INACCESSIBLE,
      message: `The path isn't a directory`,
    },
  };
}
if (!fs.existsSync(path.resolve(absPath, './index.html'))) {
  return {
    data: null,
    error: {
      type: DETECT_ERR_CODES.INACCESSIBLE,
      message: `The directory doesn't contain index.html`,
    },
  };
}

emitIpcMainEvent('__internal_main:app:cache-dapp-id-to-abspath', {
  [checkedOutDappInfo.localFSID]: checkedOutDappInfo.localFSPath,
});

let fallbackFavicon: string | undefined;
let targetMetadata: ISiteMetaData | undefined;
const { mainSession } = await getSessionInsts();
const checkResult = await checkUrlViaBrowserView(
  checkedOutDappInfo.dappURLToPrview,
  {
    session: mainSession,
    onMetaDataUpdated: (meta) => {
      fallbackFavicon = pickFavIconURLFromMeta(meta);
```

```
        targetMetadata = meta;
      },
      timeout: DFLT_TIMEOUT,
    }
  );
```

For example, on the Mac platform, if you create a symbolic link from "/etc/passwd" to "/tmp/index.html" and pass it for checking, it will not be detected as a symbolic link file.

```
ln -s /etc/passwd /tmp/index.html
```

Malicious DApps could create symbolic links and point the index.html file to sensitive files or directories, bypassing security checks.

Solution

To prevent such attacks, you can use the `isSymbolicLink()` and `isFile()` methods of the `fs.Stats` object returned by the `fs.lstatSync()` method to check if a file is a symbolic link.

Status

Fixed

Category: Others

Content

In Rabby Wallet Desktop, the "shell.openExternal()" function is used in multiple code sections to open the given external protocol URL using the default method of the desktop. However, due to the lack of URI scheme validation, any dangerous URI scheme can be directly opened, resulting in remote code execution.

For example, when handling external links in a DApp page, after checking if the protocol is "https" or "http", other protocols should be opened externally without further validation.

- src/main/streams/app.ts#line221-230

```
onIpcMainEvent(
  '__internal_rpc:app:open-external-url',
  async (evt, externalURL) => {
```

```
    const currentURL = evt.sender.getURL();
    const isFromDapp = isUrlFromDapp(currentURL);
    if (isFromDapp) return;

    shell.openExternal(externalURL);
  }
);
```

- src/isomorphic/url.ts#line154-164

```
export function isUrlFromDapp(url: string) {
  if (url.startsWith('https:')) return true;

  if (url.startsWith('http:')) {
    if (url.includes(`${LOCALIPFS_BRAND}.`)) return true;
    if (url.includes(`${LOCALFS_BRAND}.`)) return true;
    if (url.includes(`${ENS_LOCALHOST_DOMAIN}`)) return true;
  }

  return false;
}
```

Solution

It is recommended to perform strict validation of the URI scheme when using shell.openExternal().

The official security recommendations by Electron also caution against passing untrusted input to this function.

Status

Fixed

Category: Others

Content

Rabby wallet desktop does not validate the sender of all IPC messages by default.

The `ipcMain.handle` method has been rewritten as the `handleIpcMainInvoke` function.

- src/main/utils/ipcMainEvents.ts#109-117

```
export function handleIpcMainInvoke<T extends IInvokesKey = IInvokesKey>(
  eventName: T,
  handler: (
```

```
    event: Electron.IpcMainEvent,
    ...args: ChannelInvokePayload[T]['send']
  ) => ItOrItsPromise<ChannelInvokePayload[T]['response']>
) {
  ipcMain.handle(eventName, handler as any);
}
```

However, in the code, it is not found that the handleIpcMainInvoke function is used to validate the sender attribute of the incoming IPC message to ensure that operations are not executed from untrusted renderers.

For example, the handleIpcMainInvoke('dapps-put' method used in the code for saving DApp operations.

```
handleIpcMainInvoke('dapps-put', (_, dapp) => {
  // TODO: is there mutex?
  const dappsMap = dappStore.get('dappsMap');

  dappsMap[dapp.origin] = {
    ...dappsMap[dapp.origin],
    ...dapp,
  };
  dappStore.set('dappsMap', dappsMap);

  emitIpcMainEvent('__internal_main:dapps:changed', {
    dapps: getAllDapps(),
  });
});
```

Solution

According to official security recommendations, you should always validate incoming IPC messages sender property to ensure you aren't performing actions or sending information to untrusted renderers.

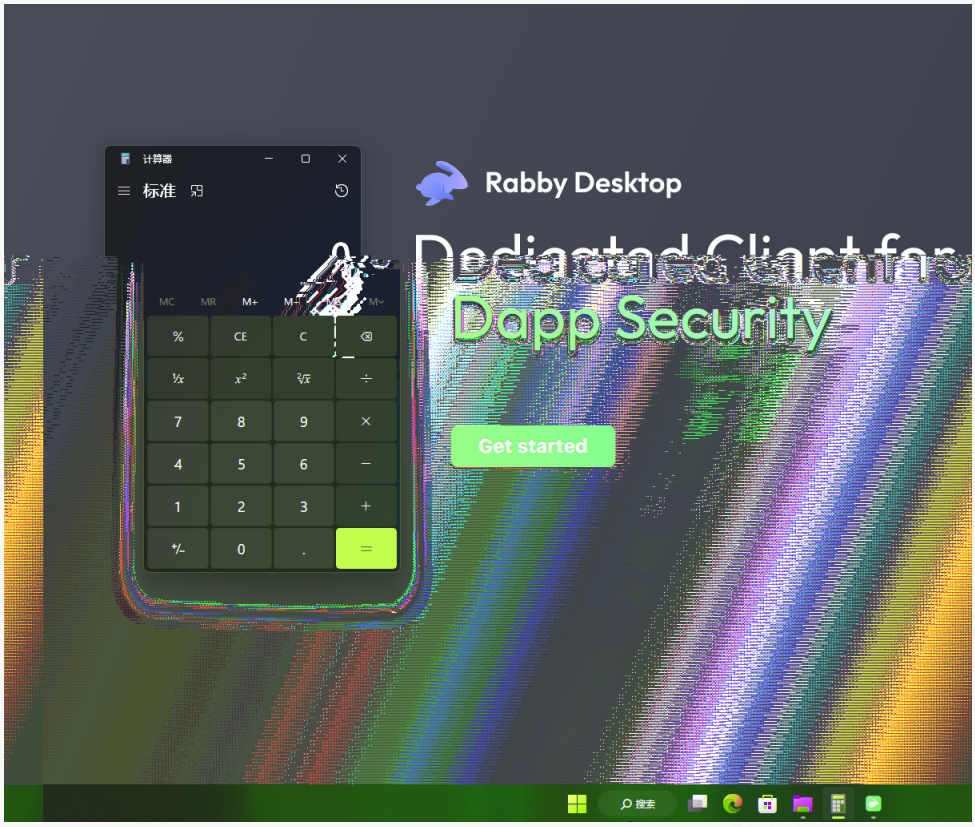
Status

Fixed

signing and opening it, triggers a "File is damaged" alert. However, in the Windows version, there is no such notification, and the modified executable program can still be opened.

For example, adding the following code to the resources file in the `app.asar` file will result in the execution of the code (the example below opens a Windows computer calculator).

```
require('child_process').execSync('c:\\windows\\system32\\calc.exe');
```



Therefore, the modified "Rabby Desktop.exe" could potentially be used for malicious phishing.

Solution

Due to Electron's support for performing integrity checks on asar files on the Windows platform, it is worth considering the option of compiling the code to V8 bytecode for mitigating and enhancing security.

Status

Confirmed

4 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002309150001	SlowMist Security Team	2023.09.07 - 2023.09.15	**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 1 high risk, 1 medium risk, 1 low risk, 1 suggestion vulnerabilities. And 1 low vulnerabilities were confirmed. All other findings were fixed. We extend our gratitude for Rabby Desktop Wallet team recognition of SlowMist and hard work and support of relevant staff.

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