Audit Report

Cypher Wallet

v1.2

May 19, 2023

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by CypherD Wallet Inc to perform a security audit of Cypher Wallet.

The objectives of the audit are as follows:

1. Determine the correct functioning of the system, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine platform integration bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repository:

https://github.com/CypherD-IO/cypherd-ui

Commit hash: `5d60ca3c0971b46de26a8d3fc4f46bc3399f86e7`

After the audit, the client has moved the audited state of the codebase with minimal modifications to the following open-source repository:

https://github.com/CypherD-IO/mobile-app

The following commit hash contains the audited state of the codebase with minimal modifications such as removal of configuration files and values and formatting rules applied to several files: `2720b14f220261fff36e2796f6b6a49ab423b055`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
    d. Weak cryptography
    e. Data leakage
    f. Password sanity
    g. Access and authorization control
4. Report preparation

Functionality Overview

Cypher Wallet is a non-custodial multi-platform (iOS, Android) mobile wallet supporting multiple EVM and Cosmos chains. It uses the React Native framework.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium	-
Level of documentation	Low	Only an architectural overview was available. Because the codebase is very large, dedicated documentation for the different components would be helpful.
Test coverage	Low	The audited codebase has no automated tests.

Summary of Findings

No	Description	Severity	Status
1	A website viewed with the wallets' web browser can place arbitrary information in browser and wallet activity history	Major	Resolved
2	The browser shows secure green lock icon next to URL even when the actual protocol is clear-text HTTP	Major	Resolved
3	It is possible to delete the wallet without providing a correct PIN	Major	Resolved
4	Sensitive data may be logged	Major	Resolved
5	Insufficient ENS name validation enables social engineering attacks	Major	Resolved
6	Sign-in requires signing an arbitrary message	Major	Resolved
7	Outdated use of Facebook Conceal as the default secure storage option on Android	Major	Resolved
8	Injected JavaScript is not verified	Major	Resolved
9	Known vulnerabilities in dependencies	Major	Acknowledged
10	The browser leaks wallet addresses to websites visited	Minor	Acknowledged
11	Users cannot set a PIN if biometrics are enabled	Minor	Acknowledged
12	The Ethereum address of a user is sent to external platforms	Minor	Acknowledged
13	Hard-coded Cosmos gas prices	Minor	Acknowledged
14	No pagination used when querying APIs with a limit	Minor	Acknowledged
15	The seed phrase is shared insecurely between view navigations	Minor	Resolved
16	PIN code length is limited to 4 digits	Minor	Acknowledged
17	Small entered token fractions are incorrectly parsed	Minor	Acknowledged
18	Number formatting is not using the users' locale	Minor	Acknowledged

19	Static retry delay of 100ms for failed API HTTP requests	Minor	Acknowledged
20	Simple RPC URL masking may reveal too little information	Informational	Acknowledged
21	Seed phrase can be screenshotted	Informational	Acknowledged
22	Limited support of 24-word mnemonics	Informational	Resolved
23	Hardcoded API keys	Informational	Resolved

Detailed Findings

1. A website viewed with the wallets' web browser can place arbitrary information in browser and wallet activity history

Severity: Major

The in-app browser allows a web page to send messages back to the application. A web-page can set any `host`, `title url`, and `origin` values which are later displayed in the browser and wallet activities histories. Moreover, a web page can set any arbitrary `URL` for its own entry in the browser history.

This functionality opens possibilities for phishing attacks – a user might unknowingly navigate to a malicious URL by clicking on entries in the browser history.

Additionally, the arbitrary URL set by a web page will be displayed in the wallet activities history if the web page has initiated web3 wallet transactions.

Recommendation

We recommend reviewing what kind of information a web page is allowed to set.

Status: Resolved

2. The browser shows secure green lock icon next to URL even when the actual protocol is clear-text HTTP

Severity: Major

The in-app Browser always shows a green lock icon next to the URL being visited even if the used protocol is insecure HTTP. This can mislead users into thinking that a connection is encrypted, although it is not. Attackers can exploit this issue for example through man-in-the-middle attacks or redirects to unencrypted versions of the site without the user noticing.

Recommendation

We recommend clearly informing the user if the HTTP protocol is being used to communicate with a website, for example by showing a warning message.

Status: Resolved

3. It is possible to delete the wallet without providing a correct PIN

Severity: Major

If an incorrect PIN is entered ten times, the application provides an option to delete the old wallet and create a new one. Since the wallet application does not require that access to a mobile device is secured by a PIN or biometric measures, a malicious actor can simply delete the wallet on a device.

Recommendation

We recommend informing the user about this mechanism (for instance before backing up the seed phrase) and require explicit confirmation by the user that they are aware of it. Additionally, we recommend making this feature optional such that users can choose to opt-in.

Status: Resolved

4. Sensitive data may be logged

Severity: Major

The app uses Sentry for error tracking without any filtering of sensitive information. Because of that, sensitive data may be sent to external servers. If an attacker manages to get access to such a server (e.g., by compromising team members or vulnerabilities in the server), they may be able to access this data. Such attacks have happened in the past. In Cypher Wallet's codebase, this could for instance happen to API tokens (in `src/core/card.ts:15`) or keystore information (in `src/core/Keychain.tsx:107`) which is included in a captured exception.

Recommendation

We recommend removing third party services that may collect sensitive data. If the client insists on using such services, we recommend filtering all sensitive information out of the exceptions before sending them to Sentry.

Status: Resolved

5. Insufficient ENS name validation enables social engineering attacks

Severity: Major

Generally, it is a good idea to only allow ASCII-characters for ENS names because the same human-readable name may have multiple valid UTF-8 encodings. Attackers can abuse this to trick users into sending funds to unintended ENS names that they have registered.

The following regex pattern is used in `src/core/util.tsx` to validate ENS names:

```
/[-a-zA-Z0-9@:%._\+~#=]{1,256
}\.[a-zA-Z0-9()]{1,6}\b([-a-zA-Z0-9()@:%_\+.~#?&//=]*)?/ig
```

This pattern matches on partial strings in the middle and does not require that the whole checked string adheres to it. `isValidEns` will therefore return true if the string contains some non-ASCII characters in the beginning, as long as there is at least one ASCII character before the dot.

Recommendation

We recommend adding a `^` in the beginning and a `$` in the end of the pattern to enforce that there is a match on the whole string.

Status: Resolved

6. Sign-in requires signing an arbitrary message

Severity: Major

When using the Cypher Wallet the first time, a sign-in to the CypherD API server is performed in the background. This sign-in requires signing a message fetched from the CypherD API server in `src/core/globalContext.tsx:153`. This is a security measure to ensure that the current user controls the wallet's public address. However, signing arbitrary messages can be harmful. For example, they can be used to steal funds from the user or perform malicious actions in the event of a compromised server.

This issue is also observed in the following locations:

- `src/containers/DebitCard/aptoCard.tsx:59`
- `src/containers/InfoScreen/legalAgreement.tsx:44`

Recommendation

We recommend crafting the message to be signed on the client side with the necessary inputs fetched from the CypherD API.

Status: Resolved

7. Outdated use of Facebook Conceal as the default secure storage option on Android

Severity: Major

The `_setInternetCredentialsOptions` function in `src/core/Keychain.tsx:92` uses Facebook Conceal as the storage for the cipher. However, for Android API level 23+, the Android Keystore is the recommended storage option.

Moreover, in the case of Android before API level 23, an encrypted ciphertext is stored in Shared Preferences together with the plain-text decryption key.

Recommendation

We recommend using the default settings provided by the `react-native-keychain` library, which selects the best available storage.

We additionally recommend either restricting the wallet app to be used only on Android API level 23+ devices or clearly informing the user about the security risks.

Status: Resolved

8. Injected JavaScript is not verified

Severity: Major

The `WebView` that is used in `src/components/WebScreen.tsx` injects the content of a JavaScript file that is loaded from the server public.cypherd.io before loading the content. However, this code is not verified in any way. An attacker that manages to control the server that is responsible for public.cypherd.io can therefore distribute malicious JavaScript that is executed before every page load.

Recommendation

We recommend using a feature like subresource integrity to verify the integrity of the downloaded file.

Status: Resolved

9. Known vulnerabilities in dependencies

Severity: Major

There are NPM packages with known vulnerabilities (which are shown when running `npm audit`). The output of `npm audit` indicates that **56 vulnerabilities** were found in the

audited packages. Of these vulnerabilities, **1** is rated as low, **14** are rated as moderate, **33** are rated as high, and **8** are rated as critical.

Recommendation

We recommend updating these dependencies.

Status: Acknowledged

10. The browser leaks wallet addresses to visited websites

Severity: Minor

The in-app browser allows a web page to send messages back to the application. A web page can request user wallet addresses and this information will be provided to the web page automatically without user consent.

Recommendation

We recommend providing user wallet address information only to whitelisted websites which have been explicitly approved by the user.

Status: Acknowledged

11. Users cannot set a PIN if biometrics are enabled

Severity: Minor

The app only requires a PIN if logging in with biometric data is not activated on the device. This may be insufficient for some use cases. For instance, a user that regularly shares their device may still want to set a PIN, even if they authenticate with their fingerprint or face scan.

Recommendation

We recommend allowing users to set a PIN irrespective of biometrics being enabled.

Status: Acknowledged

12. The Ethereum address of a user is sent to external platforms

Severity: Minor

The Ethereum address of a user is passed as the user ID to Intercom through the `Intercom.registerIdentifiedUser` function. While this may make the handling of

support requests easier, a user might not want to be associated with their Ethereum address, as this can result in a loss of anonymity.

Recommendation

We recommend using another identifier for Intercom.

Status: Acknowledged

13. Hard-coded Cosmos gas prices

Severity: Minor

In `src/constants/cosmosConfig.ts`, the gas price for every supported chain is hard-coded. This can lead to too high or too low gas prices. In the first case, a user unnecessarily pays too much for a transaction. With too low gas prices, the transaction can be delayed and may fail.

Recommendation

We recommend querying the gas price dynamically.

Status: Acknowledged

14. No pagination is used when querying APIs with a limit

Severity: Minor

The configuration in the file `src/core/globalContext.tsx` contains multiple API endpoints with a limit of 1,000. One such endpoint is `initialGlobalState.rpcEndpoints.COSMOS.otherUrls.balance`. Besides the `balances` key, this endpoint contains a key `pagination` that contains the next URL to query if the result is larger than the limit. However, this key is ignored and only the key `balances` is used in `src/core/cosmosStaking.tsx`. In rare edge cases where the queried address has a non-zero balance for over 1,000 coins, it can happen that the desired balance is not found because only the first 1,000 results are considered.

Recommendation

We recommend checking if there are more results when a limit is used and incorporating these results if there is no match in the first requested batch.

Status: Acknowledged

15. The seed phrase is shared insecurely between view navigations

Severity: Minor

In `src/containers/Options/SecurityPrivacy.tsx:58` and `src/containers/OnBoarding/confirmSeedPhrase.tsx:16`, the loaded recovery seed phrase is passed as a navigation parameter to the next screen view via the `navigate` function. This imposes a potential security issue, as the seed phrase is passed unencrypted to the next screen view and could be logged by an integrated tracking plugin (e.g., Sentry or Intercom).

Recommendation

We recommend using a secure storage mechanism (e.g., keychain) to store and retrieve the seed phrase.

Status: Resolved

16. PIN code length is limited to 4 digits

Severity: Minor

A PIN code is used to protect the wallet from unauthorized access on Android devices and as a fallback on iOS devices in case biometric authentication is unavailable. Setting a PIN code is implemented in `src/containers/PinAuthetication/setPin.tsx:34`. However, the PIN code length is limited to four digits, which is considered weak.

Recommendation

We recommend using more than four digits, preferably six or more digits, for the PIN code.

Status: Acknowledged

17. Small entered token fractions are incorrectly parsed

Severity: Minor

Entering token fractions with a very small USD value leads to displaying an inflated USD value due to the internal representation in scientific notation. For example, entering `0.00000001` MATIC tokens with a USD value of `1.2313863e-7` USD in the send token view incorrectly shows a USD value of `1.23` USD.

Recommendation

We recommend using a library such as BigNumber.js to appropriately parse numbers of arbitrary size.

Status: Acknowledged

18. Number formatting is not using the users' locale

Severity: Minor

The Cypher Wallet does not use the decimal separator of the users' current locale when entering numbers. This currently prevents entering numbers with a decimal separator other than a dot (e.g., German users may enter 1,5 instead of 1.5) and may lead to confusion. Besides misleading display issues, mobile device keyboards may have a fixed decimal separator, and users might not be able to type the required decimal separator.

Recommendation

We recommend using the users' locale to format and parse numbers.

Status: Acknowledged

19. Static retry delay of 100ms for failed API HTTP requests

Severity: Minor

In `src/core/Http.tsx`, the `axios` library is configured with the `axios-retry` plugin to retry failed HTTP requests with a static delay of 100 milliseconds and a maximum of 5 retries. If the request fails, this could lead to a large number of requests in a short period of time, as well as insufficient time for the server to recover.

Recommendation

We recommend using the exponential backoff option the `axios-retry` library provides.

Status: Acknowledged

20. Simple RPC URL masking may reveal too little information

Severity: Informational

The app contains a feature where users can review RPC endpoints. However, the masking algorithm that is used in `src/containers/Options/hostsAndRPC.tsx` only uses the first 15 characters of the URL. This may not be sufficient to verify which RPC endpoint is being

used. For example, https://weather********** is shown for Polygon, which makes it hard to deduct the whole domain of the RPC URL.

Recommendation

We recommend showing the full URL and masking out only sensitive information.

Status: Acknowledged

21. Seed phrase can be screenshotted

Severity: Informational

When creating a new wallet, the user is advised to not store the seed phrase on any internet-connected device. This is a good recommendation, but it could be enforced with a technical measure to prevent such behavior. Currently, it is possible to take screenshots of the unmasked seed phrase. In contrast, some other mobile wallets do not allow this by automatically masking the phrase when the user takes a screenshot. Because users may take screenshots (that may get backed up/synced automatically to cloud providers) to remember the phrase, this can prevent some situations where the phrase is left on an internet-connected device accidentally.

Recommendation

We recommend masking the seed phrase when taking a screenshot to force users to write it down somewhere.

Status: Acknowledged

22. Limited support of 24-word mnemonics

Severity: Informational

In `src/core/HdWallet.tsx:12` and `src/containers/OnBoarding/createSeedPhrase.tsx:26`, The Cypher Wallet creates a new seed phrase with the `generateMnemonic` function from the `react-native-bip39` library using the default 128-bit entropy setting. This results in a 12-word mnemonic, which is less secure than a 24-word mnemonic.

Moreover, while importing existing wallets with a 24-word mnemonic does work, the wallet only mentions the use of 12-word mnemonics.

Recommendation

We recommend officially supporting 12 and 24-word mnemonics for importing existing wallets and 24-word mnemonics for creating new wallets.

Status: Resolved

23. Hardcoded API keys

Severity: Informational

Hardcoding API keys directly in the source files is considered a bad practice and should be avoided. Instead, API keys should be loaded from a configuration file that is not part of the committed files in the code repository. This is to prevent accidental disclosures of the API keys.

API keys/App IDs were found in the following locations:

- `src/containers/Browser/gasHelper.ts:25`
- `src/containers/FundCardScreen/cbpay.tsx:115`

Recommendation

We recommend removing hardcoded API keys from committed files.

Status: Resolved