Audit Report

# Nym Wallet

v1.0

June 2, 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 Nym Technologies SA to perform a security audit of Nym 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/nymtech/nym/

Only the `nym-wallet` directory is in scope of this audit.

Commit hash: `e71f6f0f1538376759be560aff7f615a29a93f7a`

## 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

Nym wallet is a non-custodial multi-platform (Windows, macOS, Linux) desktop wallet for the Nym mixnet. It is written in Rust and TypeScript and uses the Tauri 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-High** | - |
| Level of documentation | **Medium** | - |
| Test coverage | **Medium** | - |

## Summary of Findings

| No | Description | Severity | Status |
|---|---|---|---|
| 1 | Password still in memory after logout | Major | Resolved |
| 2 | Mnemonic kept in memory after account creation within the wallet | Major | Resolved |
| 3 | Limited range of password special characters | Major | Resolved |
| 4 | User is forced to copy the mnemonic phrase to the clipboard | Major | Resolved |
| 5 | No confirmation of mnemonic when account is added in the wallet | Minor | Acknowledged |
| 6 | Mix node description requested via HTTP | Minor | Acknowledged |
| 7 | Known vulnerabilities in dependencies | Minor | Acknowledged |
| 8 | Inadequate password strength validation | Minor | Resolved |
| 9 | Unhandled errors thrown by asynchronous functions in React hooks | Minor | Acknowledged |
| 10 | Outdated password creation instructions | Informational | Resolved |
| 11 | No option to back up wallet data from the UI | Informational | Acknowledged |
| 12 | Use of recommended ESLint plugins | Informational | Acknowledged |
| 13 | Incomplete account name error message | Informational | Acknowledged |
| 14 | Unreachable code | Informational | Acknowledged |
| 15 | Usage of inappropriate Content Security Policy | Informational | Acknowledged |
| 16 | Outdated Tauri version | Informational | Acknowledged |
| 17 | Contract contains outstanding TODOs | Informational | Acknowledged |

# Detailed Findings

## 1. Password still in memory after logout

**Severity: Major**

Unlike the mnemonic, the password (which can be used to retrieve the mnemonic) is still visible in memory after the user is logged out. For instance, the following memory dump was taken after logging out on a Windows machine:

```
$ strings nym-wallet.DMP | grep "AAAa1"
AAAa1234!@
```

The password of the user was “AAAa1234+@” and this string is still in memory after logging out.

### Recommendation

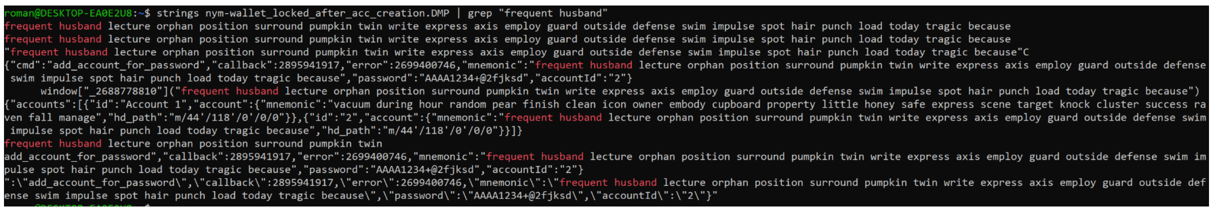
We recommend ensuring that all instances of the password in memory are deleted after logging out. Otherwise, the security measure of overwriting the mnemonic in memory can be circumvented by retrieving the password from memory and decrypting the data store file with it.

**Status: Resolved**

## 2. Mnemonic kept in memory after account creation within the wallet

**Severity: Major**

To ensure that the mnemonic is not kept in memory, it is overwritten with a fresh mnemonic in the `zeroize` implementation (within `account_data.rs`). This works well and we ensured that the mnemonic cannot be retrieved from a memory dump after the user is logged out. However, this is not the case when a new account is created within the wallet (clicking on the account symbol and then selecting “Create account”). Then, the mnemonic is still kept in memory, even after the user is logged out. This can be seen in the following screenshot, where a memory dump on Windows was taken after the user was logged out. Previously, an account with the mnemonic “frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim impulse spot hair punch load today tragic because" was added within the app:



```
roman@DESKTOP-EA0E2U8:~$ strings nym-wallet_locked_after_acc_creation.DMP | grep "frequent husband"
frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim impulse spot hair punch load today tragic because
frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim impulse spot hair punch load today tragic because
"frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim impulse spot hair punch load today tragic because"C
{"cmd":"add_account_for_password","callback":2895941917,"error":2699400746,"mnemonic":"frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense
 swim impulse spot hair punch load today tragic because","password":"AAAA1234+@2fjksd","accountId":"2"}
      window["_2688778810"]("frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim impulse spot hair punch load today tragic because")
{"accounts":[{"id":"Account 1","account":{"mnemonic":"vacuum during hour random pear finish clean icon owner embody cupboard property little honey safe express scene target knock cluster success ra
ven fall manage","hd_path":"m/44'/118'/0'/0/0"}},{"id":"2","account":{"mnemonic":"frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim
 impulse spot hair punch load today tragic because","hd_path":"m/44'/118'/0'/0/0"}}]}
frequent husband lecture orphan position surround pumpkin twin
add_account_for_password","callback":2895941917,"error":2699400746,"mnemonic":"frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside defense swim im
pulse spot hair punch load today tragic because","password":"AAAA1234+@2fjksd","accountId":"2"}
":\"add_account_for_password\",\"callback\":2895941917,\"error\":2699400746,\"mnemonic\":\"frequent husband lecture orphan position surround pumpkin twin write express axis employ guard outside def
ense swim impulse spot hair punch load today tragic because\",\"password\":\"AAAA1234+@2fjksd\",\"accountId\":\"2\"}"
```

### Recommendation

We recommend ensuring that the mnemonic is also overwritten in all memory locations after logout when a new account was added within the wallet.

**Status: Resolved**

## 3. Limited range of password special characters

**Severity: Major**

The `PasswordStrength` React component in `nym-wallet/src/pages/auth/components/password-strength.tsx` uses regex patterns to validate the strength of a password. However, the regex pattern for a "strong" password enforces a strict set of symbols, disallowing certain characters such as "}" from being used. This restriction limits the range of possible passwords and makes it easier for an attacker to successfully guess a password through brute-force methods.

### Recommendation

We recommend using a more flexible set of symbols, allowing for a broader range of passwords.

**Status: Resolved**

## 4. User is forced to copy the mnemonic phrase to the clipboard

**Severity: Major**

During account creation, the user is presented with the mnemonic phrase and is required to click the "Copy mnemonic" button, which copies the phrase to the clipboard. This practice poses a security risk as the mnemonic phrase may be accessed by other applications.

### Recommendation

We recommend removing the "Copy mnemonic" button and its associated functionality. Instead, we recommend providing clear and concise information on how to securely store and use the mnemonic phrase, including best practices and potential risks. Also, a checkbox for the user to confirm they read and understood the provided information can be added.

**Status: Resolved**

## 5. No confirmation of mnemonic when account is added in the wallet

**Severity: Minor**

When a new account is created on the start screen using “Create account”, there is no confirmation flow ensure that the user backed up/noted down the mnemonic. Instead, only the password is required to continue.

### Recommendation

We recommend implementing mnemonic confirmation whenever a new account is created to ensure that the mnemonic is backed up properly.

**Status: Acknowledged**

## 6. Mix node description requested via HTTP

**Severity: Minor**

The function `get_mix_node_description` (within `bond.rs`) requests the description of a mix node via unencrypted HTTP. While this information is only used for display purposes (limiting the impact of a MITM attack on the connection), it is still recommended to use encrypted connections for all external requests.

### Recommendation

We recommend using HTTPS to retrieve the node description.

**Status: Acknowledged**

## 7. Known vulnerabilities in dependencies

**Severity: Minor**

There are known vulnerabilities in two dependencies (crate `chrono` and `time`):

```
Crate:      chrono
Version:    0.4.19
Title:      Potential segfault in `localtime_r` invocations
Date:       2020-11-10
ID:         RUSTSEC-2020-0159
URL:        https://rustsec.org/advisories/RUSTSEC-2020-0159
Solution:   Upgrade to >=0.4.20
Dependency tree:
```

```
chrono 0.4.19
├── vergen 5.1.17
│   └── vesting-contract 1.1.0
│       ├── validator-client 0.1.0
│       │   ├── nym_wallet 1.1.1
│       │   ├── nym-wallet-types 1.0.0
│       │   │   └── nym_wallet 1.1.1
│       │   └── nym-types 1.0.0
│       │       ├── nym_wallet 1.1.1
│       │       └── nym-wallet-types 1.0.0
│       ├── nym_wallet 1.1.1
│       ├── nym-wallet-types 1.0.0
│       └── nym-types 1.0.0
└── nym_wallet 1.1.1

Crate:     time
Version:   0.1.43
Title:     Potential segfault in the time crate
Date:      2020-11-18
ID:        RUSTSEC-2020-0071
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0071
Solution:  Upgrade to >=0.2.23
Dependency tree:
time 0.1.43
└── chrono 0.4.19
    ├── vergen 5.1.17
    │   └── vesting-contract 1.1.0
    │       ├── validator-client 0.1.0
    │       │   ├── nym_wallet 1.1.1
    │       │   ├── nym-wallet-types 1.0.0
    │       │   │   └── nym_wallet 1.1.1
    │       │   └── nym-types 1.0.0
    │       │       ├── nym_wallet 1.1.1
    │       │       └── nym-wallet-types 1.0.0
    │       ├── nym_wallet 1.1.1
    │       ├── nym-wallet-types 1.0.0
    │       └── nym-types 1.0.0
    └── nym_wallet 1.1.1
```

Similarly, there are NPM packages with known vulnerabilities (which are shown when running `yarn audit`). The output of running `yarn audit` indicates that **63 vulnerabilities** were found in the audited packages. Of these vulnerabilities, **2** are rated as moderate, **46** are rated as high, and **15** are rated as critical.

### Recommendation

We recommend updating dependencies with known vulnerabilities.

**Status: Acknowledged**

## 8. Inadequate password strength validation

**Severity: Minor**

The `PasswordStrength` React component in `nym-wallet/src/pages/auth/components/password-strength.tsx` validates the strength of a password on a scale of "insufficient", "medium," and "strong". The password strength calculation is based on the length of the password and the occurrence of a set of predefined characters. However, this calculation is inadequate and may classify low entropy passwords, such as `abc1234!` or `passw0rd!` as "strong" passwords. This could lead to a complete loss of funds if an attacker can obtain an encrypted wallet file and successfully guesses the password, for example by using common password lists.

### Recommendation

We recommend using a more robust password strength calculation algorithm and excluding the use of passwords included in common password lists.

**Status: Resolved**

## 9. Unhandled errors thrown by asynchronous functions in React hooks

**Severity: Minor**

If errors thrown by asynchronous functions within React hooks are not caught and handled properly, they will be silently ignored and the React error boundary will not be triggered. The user will not be notified of the error and the application may end up in an inconsistent state.

The following but incomplete list of hooks is affected:

- `nym-wallet/src/context/main.tsx:109`
- `nym-wallet/src/context/main.tsx:147`
- `nym-wallet/src/context/main.tsx:160`

### Recommendation

We recommend explicitly catching potential errors thrown by `async` functions and, if necessary, using the `useErrorHandler` hook from the `react-error-boundary` package to trigger the React error boundary.

**Status: Acknowledged**

## 10.Outdated password creation instructions

**Severity: Informational**

The password creation process for the wallet includes a series of outdated steps that do not accurately reflect the current version of the wallet application. This can cause confusion for users and may result in them being unable to create a password for their account.

Specifically, the third step instructs the user, “*On the next screen select ‘Create a password for your account’*”, but instead, the create password button is located on the login with mnemonic screen and labeled "Create a password".

#### Recommendation

We recommend updating the steps to reflect the current version of the wallet application.

**Status: Resolved**

### 11. No option to back up wallet data from the UI

**Severity: Informational**

There is currently no way to back up the `user-data.json` file within the UI of the wallet. If a user wants to do so (e.g., for backup purposes), they needs to find and copy the file manually, which may be difficult for technically less sophisticated users.

#### Recommendation

We recommend adding an option to the UI to download the file directly.

**Status: Acknowledged**

### 12. Use of recommended ESLint plugins

**Severity: Informational**

ESLint is a tool used for statically analyzing JavaScript code to identify and prevent potential errors and security vulnerabilities. Nym Wallet extends the ESLint configuration provided by the `@nymproject/eslint-config-react-typescript` package. However, the configuration does not include recommended security and React hooks plugins. This could lead to missing potential security vulnerabilities and bugs.

### Recommendation

We recommend using the following ESLint plugins in addition to the current plugins and fixing any issues that arise:

- `plugin:security/recommended`
- `plugin:react-hooks/recommended`

**Status: Acknowledged**

## 13. Incomplete account name error message

**Severity: Informational**

The error message displayed by the `NameAccount` React component in `nym-wallet/src/components/Accounts/modals/AddAccountModal.tsx:107` when an invalid account name is entered does not mention spaces as a valid character, despite being allowed in the account name. This can cause confusion for users and may result in the creation of undesirable account names.

### Recommendation

We recommend adapting the error message to include spaces as valid characters.

**Status: Acknowledged**

## 14. Unreachable code

**Severity: Informational**

The `NodeStatus` React component contains an unreachable `return` statement. This statement is unreachable because the `switch` statement's `default` case handles all possible values of the `status` variable besides the ones that are explicitly handled by the other cases.

### Recommendation

We recommend removing the unreachable `return` statement.

**Status: Acknowledged**

## 15. Usage of inappropriate Content Security Policy

**Severity: Informational**

In `nym-wallet/src-tauri/tauri.conf.json`, a very permissive Content Security Policy (CSP) is used that allows remote content and scripts, as well as (unsafe) inline script/eval. Usage of insecure CSP values during development is commonly accepted, however, other application developers take the published CSPs as examples, which could lead to insecure apps in the wild.

### Recommendation

We recommend restricting CSP configuration to allow a minimal set of protocols and sources for each protocol.

**Status: Acknowledged**

## 16. Outdated Tauri version

**Severity: Informational**

There is a known vulnerability in Tauri 1.04 curently in the configuration file `nym-wallet/src-tauri/Cargo.toml`. In versions prior to 1.0.7 and 1.1.2, Tauri is vulnerable to an Incorrectly-Resolved Name. Due to incorrect escaping of special characters in paths selected via the file dialog and drag and drop functionality, it is possible to partially bypass the `fs` scope definition.

Reference: CVE-2022-41874

### Recommendation

We recommend upgrading versions to 1.0.7, 1.1.2, and 1.2.0. As a workaround, disable the dialog and `fileDropEnabled` component inside `tauri.conf.json`.

**Status: Acknowledged**

## 17. Contract contains outstanding TODOs

**Severity: Informational**

The contracts within the codebase contain outstanding TODO items in the following paths:

- `nym-wallet/nym-wallet-types/src/interval.rs:7`
- `nym-wallet/src/components/Accounts/AccountItem.tsx:62-72`
- `nym-wallet/src/components/Accounts/MultiAccountHowTo.tsx:15`
- `nym-wallet/src/components/Accounts/MultiAccountWithPwdHowTo.tsx:14`
- `nym-wallet/src/components/Bonding/BondedMixnode.tsx:117`
- `nym-wallet/src/components/Bonding/modals/NodeSettingsModal.tsx:57`

- `nym-wallet/src/components/Delegation/types.ts:7`
- `nym-wallet/src/context/bonding.tsx:79,400,435`
- `nym-wallet/src/pages/bonding/node-settings/settings-pages/NodeUnbondPage.tsx:16`
- `nym-wallet/src/pages/bonding/node-settings/settings-pages/general-settings/index.tsx:12`
- `nym-wallet/src/pages/settings/system-variables.tsx:104,110`
- `nym-wallet/src-tauri/src/config/mod.rs:51`
- `nym-wallet/src-tauri/src/operations/mixnet/rewards.rs:14`
- `nym-wallet/src-tauri/src/operations/simulate/cosmos.rs:26`
- `nym-wallet/src-tauri/src/operations/simulate/cosmos.rs:26`

### Recommendation

We recommend completing all outstanding TODO items.

**Status: Acknowledged**