Tezoro Snap

Date	April 2024
Auditors	Valentin Quelquejay

1 Executive Summary

This report presents the results of our engagement with **Tezoro** to review the **Tezoro Snap**.

The review was conducted over five days, from **April 8, 2024** to **April 12, 2024**, by **Valentin Quelquejay**. A total of 5 person-days were spent.

The Tezoro snap is designed to communicate with the Tezoro dApp. It is an extension of the main dashboard. Specifically, it monitors user's token balances every fifteen days and alerts them when their token balance exceeds a certain treshold, suggesting them to backup their assets to Tezoro if they haven't already done so. Users can link the snap with their Tezoro account via the Tezoro dashboard.

2 Scope

Our review focused on the commit hash 4204075301856f7412f07746c937343fb31cb7b8. The list of files in scope can be found in the Appendix.

2.1 Objectives

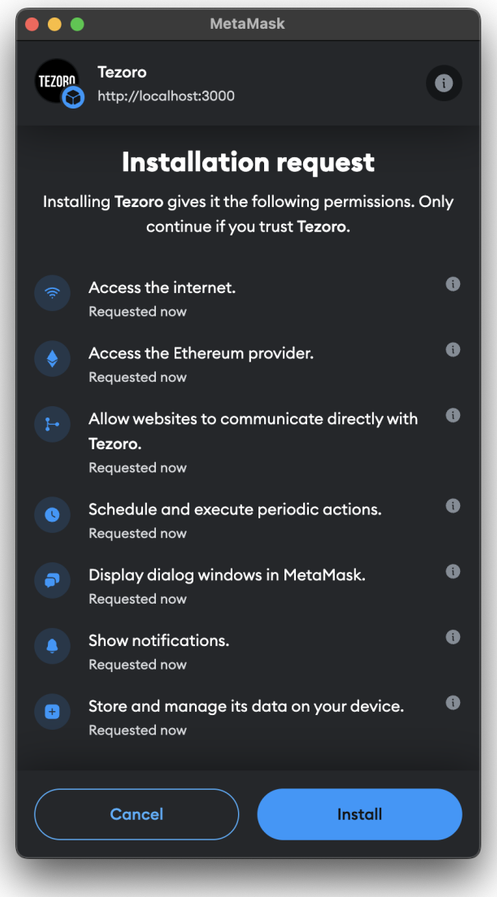
Together with the **Tezoro** team, we identified the following priorities for our review:

1. Correctness of the implementation, consistent with the intended functionality and without unintended edge cases.
2. Identify vulnerabilities particular to the MetaMask Snaps SDK integration in coherence with the MetaMask Snap Threat Model describing a Snap as an extension of the MetaMask Wallet Trust Module.

3 Snap Outline

- The snap communicates and authenticates with the Tezoro API using the user's token provided by the companion dApp.
- Connected dApps can communicate with the snap via RPC calls.
- The snap triggers a cron job every fifteen days to notify the user about tokens that are not backed up.
- The snap stores data in persistent storage.

3.1 Capabilities



Details

```
🚜 [snap_dialog]
    ☝️ - snap_dialog - Displays a dialog in the MetaMask UI. There are three types of dialogs with different parameters and return types.
    ⚠️ - this method renders Markdown! check for ctrlchar/markdown/injection
       🔶 src/index.ts
🚜 [snap_notify]
    👌 - snap_notify - Displays a notification in MetaMask or natively in the browser. Snaps can trigger a short notification text for actionable or ti
     ⚠️ - this method renders Markdown! check for ctrlchar/markdown/injection
       🔶 src/index.ts
🚜 [snap_manageState]
    🪶 - snap_manageState - snap can store up to 100mb (isolated)
       🔶 src/index.ts
       🔶 src/check-tokens.ts
🚜 [endowment:network-access]
    🌐 - endowment:network-access - snap can access internet
     ⚠️ - this method may leak information to external api
       🔶 src/get-backups.ts
       🔶 src/external/get-price-of-asset-quoted-in-usd.ts
🚜 [endowment:ethereum-provider]
    🔺 - endowment:ethereum-provider - snap can access ethereum API
    ⚠️ - check if the **snap code** (not site) actually accesses the global 'ethereum' object
see https://docs.metamask.io/snaps/learn/about-snaps/apis/#snap-requests
       🔶 src/index.ts
       🔶 src/check-tokens.ts
🚜 [endowment:rpc]
    ❗ - endowment:rpc.dapps - snap can communicate with websites/dapps; check origin for internal api calls!
       🔶 src/index.ts
🚜 [endowment:cronjob]
    - Cron Job 0:
      👉 At 12:00 AM, on day 1 and 15 of the month
     used in:
      🔶 src/index.ts
🎄 - Package Dependencies:
     - @metamask/rpc-errors:^6.2.1              (🔺 looks like devDependency 👀)
     - @metamask/snaps-sdk:^3.1.0               (🔺 looks like devDependency 👀)
     - buffer:^6.0.3           (🔺 looks like devDependency 👀)
     - viem:^2.7.22            (🔺 looks like devDependency 👀)
     - zod:^3.22.4             (🔺 looks like devDependency 👀)
```

4 Findings

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

4.1 Lack of Origin Check on RPC Requests Critical ✓ Fixed

Resolution
Addressed by tezoroproject/metamask-snap#41

Description

The Snap does not validate the origin of RPC requests, allowing any arbitrary dApp to connect to the Snap and initiate arbitrary RPC requests. Specifically, any dApp can access the privileged getToken and deleteToken RPC endpoints. Consequently, a malicious dApp could potentially extract a user's Tezoro token from the Snap and impersonate the user in interactions with the Tezoro API. Depending on the permissions associated with this token, the implications could be critical.

Example

packages/snap/src/index.ts:L14-L18

```
export const onRpcRequest: OnRpcRequestHandler = async ({ request }) => {
  switch (request.method) {
    case 'requestAccounts': {
      const data = await ethereum.request({
        method: 'eth_requestAccounts',
```

packages/snap/src/index.ts:L64-L65

```
case 'getToken': {
  const state = await snap.request({
```

packages/snap/src/index.ts:L34-L35

```
case 'saveToken': {
  const result = await snap.request({
```

Recommendation

Validate the origin of all incoming RPC requests. Specifically, restrict access to the RPC endpoints to only the Tezoro management dApp. Additionally, consider removing any endpoints that are not essential for the Snap's functionality. For example, the `getToken` endpoint for extracting the API token might be unnecessary and could be removed to enhance security.

4.2 `getPriceOfAssetQuotedInUSD` Might Return Flawed Asset Prices Major ✓ Fixed

Resolution
Addressed by tezoroproject/metamask-snap#42

Description

First, the function `getPriceOfAssetQuotedInUSD()` operates under the assumption that stablecoins—specifically 'USDT', 'USDC', 'DAI', 'USDP', and 'TUSD'—always maintain a 1:1 price ratio with the USD. Although this is generally expected to be the case, there have been instances where some stablecoins failed to uphold their peg to the USD. In such scenarios, this assumption no longer holds true, resulting in the return of inaccurate balances. Furthermore, it's important to note that the prices returned by this function are quoted in USDT, despite the function's name suggesting that prices are returned in USD. This could lead to discrepancies if 'USDT' diverges from its fiat counterpart.

Second, The function `getPriceOfAssetQuotedInUSD()` assumes that every token name that starts with 'W' is a wrapped token. Thus, the initial 'W' is removed from the token name before fetching the prices from Binance API. As a result, the subsequent API request made to get the price of the unwrapped token could potentially fail or return an incorrect price, if the token name starts with a 'W' but the token is not a wrapped token. For instance, the "WOO" token is present in the list of tokens supported by the Snap. In that case, the price API will error as it will try to fetch the price of the `OOUSDT` pair instead of `WOOUSDT`.

Finally, relying on an hardcoded external APIs is sub-optimal. Indeed, it may be that the API may fail, start returning incorrect data, or simply become outdated and stop working.

Example

packages/snap/src/external/get-price-of-asset-quoted-in-usd.ts:L15-L19

```
if (assetName.startsWith('W')) {
  // Assume this is a wrapped token
  assetName = assetName.slice(1); // remove W
}
try {
```

packages/snap/src/external/get-price-of-asset-quoted-in-usd.ts:L20-L23

```
const response = await fetch(
  `https://api.binance.com/api/v3/ticker/price?symbol=${assetName.toUpperCase()}USDT`,
);
const json = await response.json();
```

Recommendation

To mitigate this issue, one should avoid making assumptions about token names. Instead, one would ideally fetch token metadata from a trusted source to determine whether a token is wrapped or not, hardcode this information in the token-list, or directly fetch the price of the wrapped token.

Moreover, instead of hardcoding the price API, we would recommend setting up a custom API Gateway which provides a layer of abstraction between the Snap and the external APIs it uses. This would provide flexibility and allow quickly swapping for other external APIs in case they stop behaving properly.

4.3 Inaccurate Return Value in `checkTokens()` Medium ✓ Fixed

Resolution
Addressed by tezoroproject/metamask-snap#43

Description

The function `checkTokens()` checks if `token` exists in `parsedState.data` and returns `{isStatePresent: true, isTokenPresent: true,}` if it does not. This is incoherent as `isTokenPresent` should be false in that case.

Examples

packages/snap/src/check-tokens.ts:L41-L46

```
if (!token) {
  return {
    isStatePresent: true,
    isTokenPresent: true,
  };
}
```

packages/snap/src/schemas.ts:L35-L37

```
export const stateSchema = z.object({
  token: z.string().optional(),
});
```

Recommendation

Fix the return value. `isTokenPresent` should be false if the token is not present in the state. Alternatively, fix the zod `stateSchema` to ensure that `token` is not optional. In that case the `safeParse` function will fail and the function will return the correct value.

4.4 cronJob `checkTokens` Might Flood User Notifications Medium ✓ Fixed

Resolution
Addressed by tezoroproject/metamask-snap#44. The snap now sends a single notification.

Description

The snap includes a cron job named `checkToken` that activates every 15 days to verify which user tokens are backed up and which are not. For each token identified as not backed up (listed in `tokenList`), the snap issues a notification to the user. If the list of unbacked tokens is extensive, the user will receive many notifications, potentially undermining the effectiveness of these alerts or causing the user to overlook other important notifications. To alleviate this concern, it is recommended to aggregate these notifications. Issuing a single notification, or capping the number of notifications when the size of `tokenList` surpasses a specific threshold (e.g., 5), could improve the user experience.

Examples

packages/snap/src/index.ts:L122-L125

```
[...tokensList].map(async (token) => {
  await snap.request({
    method: 'snap_notify',
    params: {
```

Recommendation

We would recommend to aggregate notifications, summarizing the status of unbacked tokens, at least when their number exceeds a certain reasonable threshold.

4.5 `deleteToken` Should Prompt User for Its Consent Medium ✓ Fixed

Resolution
Addressed by tezoroproject/metamask-snap#45

Description

As a rule of thumb, every state-changing interaction with the Snap's state should require user confirmation, and the process should be aborted if the user does not consent. This principle is already applied to the `saveToken` RPC endpoint. To maintain consistency and ensure user control over their data, the `deleteToken` endpoint should also prompt the user for consent before proceeding to delete the token from the Snap's state.

Examples

packages/snap/src/index.ts:L85-L96

```
case 'deleteToken': {
  await snap.request({
    method: 'snap_manageState',
    params: {
      operation: ManageStateOperation.UpdateState,
      newState: {},
      encrypted: true,
    },
  });
  return true;
}
```

Recommendation

Similarly to the `saveToken` RPC endpoint, the `deleteToken` endpoint should ask the user for its consent before deleting the token from the snap's state.

4.6 cronJob `checkTokens` Return Value Is Not Necessary Minor ✓ Fixed

Resolution

Addressed by tezoroproject/metamask-snap#46

Description

The cronJob `checkTokens` return value is not necessary as it will never be accessed, and should be omitted.

Examples

packages/snap/src/index.ts:L133-L136

```
return {
  data,
  error,
};
```

Recommendation

Drop the return value

4.7 Potential Markdown Injection in `snap_notify` Minor ✓ Fixed

Resolution

Fixed by addressing issue issue 4.4

Description

It should be noted that the `snap_notify` message is not protected against markdown injection. This vulnerability means that token names could potentially be used to inject malicious characters into the prompt. Since token names are currently sourced from a predefined list of supported tokens, the risk is mitigated for the time being. However, it is important to consider this vulnerability, especially if the list of supported tokens is expanded or modified in the future.

Examples

packages/snap/src/index.ts:L122-L130

```
[...tokensList].map(async (token) => {
  await snap.request({
    method: 'snap_notify',
    params: {
      type: 'native',
      message: `Protect ${token} from loss with on-chain backup & will`,
    },
  });
});
```

Recommendation

Sanitize the token names to protect against markdown injections.

4.8 External/User Input Sanitization Minor ✓ Fixed

Resolution

Fixed in tezoroproject/metamask-snap#47

Description

It is important that every external or user input is validated to protect against injection vulnerabilities. While the zod library is utilized for validation in most instances within the codebase, there are exceptions where external inputs are not sanitized. This oversight could lead to potential security vulnerabilities.

Examples

packages/snap/src/index.ts:L46-L61

```
  const { params } = request;

  await snap.request({
    method: 'snap_manageState',

    params: {
      operation: ManageStateOperation.UpdateState,
      newState: {
        token: params.token,
      },
      encrypted: true,
    },
  });
  return true;
}
return false;
```

Recommendation

To mitigate potential security risks, make sure to implement comprehensive input validation for all untrusted inputs across the entire codebase. Specifically, for the example provided, utilizing zod to sanitize `params.token` and throw if the token does not adhere to the expected format would help in preventing bugs and potential injection attacks. Establishing a consistent validation practice will help prevent vulnerabilities related to unsanitized inputs.

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
packages/snap/jest.config.js	`06eeb5e61e820fa8b33acaa02daeb5f45e2624bd`
packages/snap/snap.config.ts	`cdd2b09e283e57825ba279dd5ae27771dbc47362`
packages/snap/src/abi/ERC20.ts	`649a098160c9e5c5f21b54f7a10806acff93590f`
packages/snap/src/abi/Tezoro.ts	`11f6f19cf4132578be2a491b5f82edaa50f45011`
packages/snap/src/check-tokens.ts	`a402a78fa0fe1cb567a26cfa8737283198642fd6`
packages/snap/src/constants.ts	`66053e7238946ebb9dc64fedafe83d172b2af522`
packages/snap/src/external/get-price-of-asset-quoted-in-usd.ts	`043e7c8c5c22973a2f0aab317f2071208e0d1f55`
packages/snap/src/get-active-backups.ts	`439b3d8fd2d57b80bf4316066916d7e646f94f4d`
packages/snap/src/get-backups.ts	`77cc64d1f3aad7d6ed62d9323fefcc839d09106a`
packages/snap/src/get-token-balances.ts	`b5b3e7189a2a595bb7af19abf73a961800e1d1ca`
packages/snap/src/index.test.ts	`fc255067fc212f97b291f7c215f3c281d31d4ff1`
packages/snap/src/index.ts	`38b674508aad086de11f85736efadab10f3f0dba`
packages/snap/src/public-client.ts	`0d201354d6674704c0a93bf7e00283bd091de028`
packages/snap/src/schemas.ts	`9594244efeaf8becbe32bfbe40eab82411a1dcdc`
packages/snap/src/tokens-list.ts	`ca01560a7936e1f3addee41143737cc1330084d3`
packages/snap/src/types.ts	`4a611b9b889084f5d300527d5eb7a4fcb62f6aeb`
packages/snap/src/utils/assert-is-with-message.ts	`8704873a53ede576c3b822cd24867df23860d4e9`

Appendix 2 - Disclosure

Consensys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via Consensys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any third party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any third party by virtue of publishing these Reports.

A.2.1 Purpose of Reports

The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

A.2.2 Links to Other Web Sites from This Web Site

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Consensys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that Consensys and CD are not responsible for the content or operation of such Web sites, and that Consensys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that Consensys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. Consensys and CD assumes no responsibility for the use of third-party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

A.2.3 Timeliness of Content

The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice unless indicated otherwise, by Consensys and CD.