MetaMask/Partner Snaps - FilSnap

Date	August 2023
Auditors	Dominik Muhs

1 Executive Summary

This report presents the results of our engagement with **Protocol Labs** to review their **FilSnap** MetaMask Snap.

The review was conducted over two weeks, from **August 14, 2023,** to **August 18, 2023**, by **Dominik Muhs**. A total of five person-days were spent.

Most notably, two critical issues have been identified regarding displaying Markdown-rendered dialogues and the programmatic handling of private key information. Furthermore, two major findings present security considerations for the configuration functionality and the user signature confirmation dialogue lacking origin information.

A follow-up assessment has been conducted from **October 2, 2023** to **October 3, 2023** by **Dominik Muhs**. A total of two person-days were spent.

2 Scope

Our review initially focused on the commit hash `5c5e6ad523d198d12bdd2057028dad429435bc6f`. In the follow-up assessment, the fixes were verified against the commit hash `8374e6aea59941d98bf29ca5e902df9eebf919f0`. The list of files in scope for both engagements can be found in the Appendix.

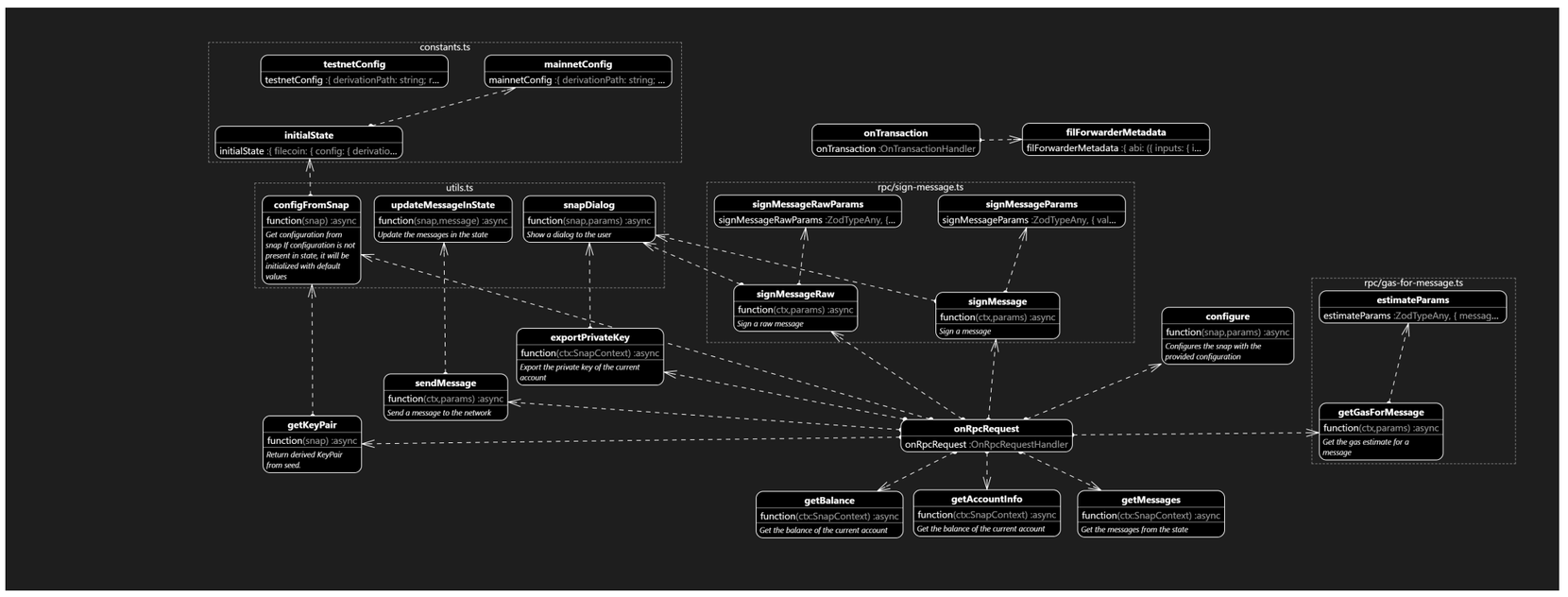
One day after the scheduled verification deadline, the development team submitted commit `6ddb227738ed3aa041c18131eee65a98e17acdf4`, fixing issue 4.4 completely. This change has been added to the report.

2.1 Objectives

Together with the **Protocol Labs** 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 System Overview



Snap Architecture (August 2023)

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 Markdown and Control Character Injection Critical ✓ Fixed

Resolution

Configuration values reflected in the confirmation prompt are implicitly validated by getting user consent on config changes through the `fil_configure` RPC call. Furthermore, confirmation dialogue values have been (mostly) displayed using the `copyable` function, resulting in escaped output.

This issue has been addressed in revision `1a8715f42cfc9f721e8faab8a7a2610f53592f94`.

Description

The snap uses MetaMask's Snaps UI package to present dialogs to users for data verification and action confirmations. While these dialogs ensure dapps don't silently execute operations without user consent, some UI components have vulnerabilities. For instance, the `text()` component can render Markdown or be susceptible to control character injections. Specifically, in FilSnap's context, when users are prompted to sign a message showing a gas cost estimate if the message contains Markdown-renderable text, the user might unintentionally sign an inaccurate message. It's critical to note that the variable `ctx.config` in the provided code snippet could contain untrusted data, potentially altering the context of the displayed message. Malicious manipulation of the snap context is outlined in issue 4.3.

packages/snap/src/rpc/sign-message.ts:L68-L89

```
const conf = await snapDialog(ctx.snap, {
  type: 'confirmation',
  content: panel([
    heading(`Send ${Token.fromAttoFIL(message.value).toFIL().toString()} to`),
    copyable(message.to),
    divider(),
    heading('Details'),
    text(
      `Gas _(estimated)_: **${gas.toFIL().toFormat({
        decimalPlaces: ctx.config.unit?.decimals,
        suffix: ` ${ctx.config.unit?.symbol}`,
      })}**`
    ),
    text(
      `Total _(amount + gas)_: **${total.toFIL().toFormat({
        decimalPlaces: ctx.config.unit?.decimals,
        suffix: ` ${ctx.config.unit?.symbol}`,
      })}**`
    ),
  ]),
})
```

Recommendation

Prioritize input validation. Encode data securely when presenting it to users. Display original data within an escaped pre-text or code block that prevents rendering non-standard characters or Markdown. Subsequently, add any derived or decoded data to offer users a comprehensive understanding.

4.2 Directly Exposed Private Key Export Critical ✓ Fixed

Resolution

While the `exportPrivateKey` function is still present and exposed via the `fil_exportPrivateKey` RPC call, it now displays a warning and only surfaces the user's private key in a secured dialogue. This is analogous to MetaMask's approach and will make users consider their actions carefully. Automated attacks without user intervention are no longer possible since users must copy-paste their keys.

This issue has been addressed in commit `c88a9ee1359e9a35735ce5d7b18b4cfcd2de0326`.

Description

The snap can access the BIP44 entropy for Filecoin's private keys, granting it considerable power over MetaMask's private keys. Specifically, the `fil_exportPrivateKey` command lets dapps obtain the private key programmatically, pending user consent. However, there's a heightened risk of users indiscriminately granting this permission. To maintain MetaMask's security integrity, the snap should mirror the same rigorous security standards to mitigate the effect of phishing attacks and malicious dapps.

packages/snap/src/rpc/export-private-key.ts:L19-L34

```
export async function exportPrivateKey(
  ctx: SnapContext
): Promise<ExportPrivateKeyResponse> {
  const conf = await snapDialog(ctx.snap, {
    type: 'confirmation',
    content: panel([heading(`Do you want to export your private key?`)]),
  })

  if (conf) {
    return {
      result: base64pad.encode(ctx.account.privateKey),
      error: null,
    }
  }
  return serializeError('User denied private key export')
}
```

Recommendation

The development team should reconsider providing such a sensitive functionality. Instead of programmatically exposing the private key, use a dialog that prompts users to copy the private key manually. This approach, consistent with MetaMask's default, encourages users to deliberate their actions more thoroughly.

4.3 `fil_configure` Allows Anyone to Change the Snap's Configuration Major Partially Addressed

Resolution

This issue has been partially addressed and practically fixed. While it is still possible for any site to change the snap's configuration, its values are now maintained under the origin site's namespace. Furthermore, when connecting to a site, the user must manually confirm the proposed configuration values.

This issue has been fixed in `1a8715f42cfc9f721e8faab8a7a2610f53592f94`.

Description

The `fil_configure` command, accessible by any dapp, accepts parameters of type `ConfigureParams`. This allows for modifying key configuration details like the derivation path, RPC details, network state, and unit data:

```
type ConfigureParams = {
    derivationPath?: string | undefined;
    rpc?: {
        url: string;
        token: string;
    } | undefined;
    network?: "mainnet" | "testnet" | undefined;
    unit?: {
        symbol: string;
        decimals: number;
        image?: string | undefined;
        customViewUrl?: string | undefined;
    } | undefined;
}
```

An attacker exploiting this open access can manipulate elements such as the network state or RPC URL, which could mislead users when signing messages. **Furthermore, given the shared instance nature of the snap across multiple pages, a malicious dapp can influence the behavior of the snap in other dapps.**

The MetaMask Snaps team has responded to this particular issue by stating that they will soon add a mutex to the SDKs state object to mitigate the problem partially. Nonetheless, even with a mutex in place, it's important to remember that business logic can still execute in an outdated state.

Recommendation

Evaluate the necessity of exposing specific configuration options through `fil_configure`. If it doesn't offer significant utility, consider removing it entirely, favoring a hardcoded configuration approach. This would ensure that critical token data and RPC URLs are modifiable only through controlled snap updates.

4.4 Lack of Signature Dialog Context and RPC Origin Major ✓ Fixed

Resolution

The signature UI dialogue now displays the user account used for signing and the origin site that triggered the process. Furthermore, details such as the RPC URL and the network name are displayed. This issue has been fully addressed in `1a8715f42cfc9f721e8faab8a7a2610f53592f94` and `6ddb227738ed3aa041c18131eee65a98e17acdf4`.

Description

The FilSnap signing dialog doesn't indicate which user account is used for signing. This omission can be exploited by malicious dapps, leading users to believe they're signing with one account when in fact, another is being used. Such transparency gaps, especially in an environment integrating multiple dapps and accounts, can mislead users into providing unintended signatures.

For reference, MetaMask displays these details during its signing requests, a practice FilSnap should adopt.

packages/snap/src/rpc/sign-message.ts:L68-L88

```
const conf = await snapDialog(ctx.snap, {
  type: 'confirmation',
  content: panel([
    heading(`Send ${Token.fromAttoFIL(message.value).toFIL().toString()} to`),
    copyable(message.to),
    divider(),
    heading('Details'),
    text(
      `Gas _(estimated)_: **${gas.toFIL().toFormat({
        decimalPlaces: ctx.config.unit?.decimals,
        suffix: ` ${ctx.config.unit?.symbol}`,
      })}**`
    ),
    text(
      `Total _(amount + gas)_: **${total.toFIL().toFormat({
        decimalPlaces: ctx.config.unit?.decimals,
        suffix: ` ${ctx.config.unit?.symbol}`,
      })}**`
    ),
  ]),
})
```

Recommendation

Display the user account used for signing and the origin domain of the RPC call during the signing dialog, ensuring users have full context before confirming any transactions.

4.5 Missing Address Protection Medium ✓ Fixed

Resolution

Address protection has been introduced with commit `1a8715f42cfc9f721e8faab8a7a2610f53592f94`. Before an origin site can access the snap's RPC calls, it has to call the `fil_configure` RPC call, which requires manual user confirmation of the connection.

Description

While MetaMask hides wallet addresses by default, requiring users to expose them to dapps manually, the snap's `fil_getAddress` and `fil_getAccountInfo` RPC endpoints always disclose the current address to any connected dapp, even if that address has not been connected to the page. This allows potentially untrusted dapps to silently retrieve all user addresses, bypassing MetaMask's intentional security design.

Recommendation

Adopt security protocols similar to MetaMask's main wallet. Let users select which addresses they share with dapps and prevent automatic exposure of non-allowlisted wallet addresses without explicit user permission.

4.6 Missing Timeout in `RPC.call` Medium ✓ Fixed

Resolution

A timeout option has been added to the `iso-filecoin` dependency in `e1908f9ea05e309c7f1d260ecc18584503155cb4` addressing this issue.

Description

Within the dependency `iso-filecoin`, there's an oversight in the RPC class. Specifically, the `call` method lacks a timeout parameter. Due to this missing timeout, the snap execution can experience delays, which could lead to an aborted request. The code excerpt provided shows the missing timeout in the fetch call.

```
const res = await this.fetch(this.api, {
  method: 'POST',
  headers: this.headers,
  body: JSON.stringify({
    jsonrpc: '2.0',
    method,
    params,
    id: 1,
  }),
})
```

Recommendation

To mitigate potential delays and ensure smoother operation, it's recommended to introduce a timeout to this `call` method. Implementing a timeout can prevent prolonged waits and enhance the resilience of the method against unexpected delays.

4.7 Unnecessary Distribution of Private Key Information Minor ✓ Fixed

Resolution

In commit `1a8715f42cfc9f721e8faab8a7a2610f53592f94` the `account` field has been removed from the snap context and is only exposed selectively in RPC handlers that require user account data. These handlers all explicitly call the `getAccount` function.

Description

The snap context includes an `account` field containing the private key. This context houses sensitive data and is distributed across all request handlers. Due to this setup, redundant access to the private key is provided to several functions, e.g.:

- `fil_getAccountInfo`,
- `fil_getBalance`,
- `fil_getMessages`,
- `fil_sendMessage`, and
- `fil_getGasForMessage`.

packages/snap/src/index.ts:L42-L47

```
const context: SnapContext = {
  config,
  rpc,
  account,
  snap,
}
```

Recommendation

It's essential to ensure the security and privacy of critical user data. As such, access to private key data should be restricted, particularly for handler functions. Adjust the implementation by adding another `SnapContext` type that holds the account's private key information and only pass it to handler functions that immediately need to handle it. Alternatively, add a lazy-loading object into the existing `SnapContext` that only loads the private key when called by an authorized function.

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
packages/adapter-react/src/index.js	`7cb10cd1a5f2846ec5d93a9af9d1f0b8f40046fb`
packages/adapter-react/src/types.ts	`f5d4de1f30866a08d6717e02bb3c23d3aaef3b58`
packages/adapter/playwright.config.js	`127128e1dd77c43d38ade5987bda3f8bc12dfe41`
packages/adapter/src/index.ts	`ec58446915c69cd542545e3f482a86e7fe318b1c`
packages/adapter/src/snap.ts	`cf03e24ea5837de305cad98e353b21eae598a6ff`
packages/snap/filecoin-logo.svg	`373e116f01cacb07ea72f87fdb59bb5a877c56f1`
packages/snap/playwright.config.js	`d151c3d356214691ce588ebb097e99d58977157c`
packages/snap/rollup.config.js	`2d8f81146be8fcbd347c1fefff22a58510a3b10a`
packages/snap/src/constants.ts	`ad936358af4539e673d11ba3ecf6f9bdfda28e1f`
packages/snap/src/filforwarder-metadata.ts	`011922ebd42f81914ce14bee35fa3cca7b1a99a4`
packages/snap/src/index.ts	`26c449212b5016a4ccf396152f9cc407f6d62ccc`
packages/snap/src/keypair.ts	`fece5b1dfdb051b67f7c46f8848d123f6e631efd`
packages/snap/src/rpc/configure.ts	`ce9335e4c4957f2d62617e7c5a8d1611768f3cc5`
packages/snap/src/rpc/export-private-key.ts	`06424760aacf6d2d79feac05112a29738f3a758f`
packages/snap/src/rpc/gas-for-message.ts	`f89ac0f4011b5317f99b504e8fd2998ee2bc4283`
packages/snap/src/rpc/get-account.ts	`75a512fc96208a3d48e836d3f5060e0333cbdcf6`
packages/snap/src/rpc/get-balance.ts	`75f8816f6764209cf854406973c6b45dfb0467d7`
packages/snap/src/rpc/get-messages.ts	`16464c8d665a0f95e5d1d64a4c561f2cd3cbb70f`
packages/snap/src/rpc/send-message.ts	`e2781fd63f39755fb8b31d1e31919142064e4a00`
packages/snap/src/rpc/sign-message.ts	`9993ecccb8a960f3635b583cac881a78da0b4505`
packages/snap/src/schemas.ts	`922e0f796ef01f65a359165c70299895e519f3bd`
packages/snap/src/transaction-insight.ts	`d09b6e69a7655e98de2309b02e7798914e87dd23`
packages/snap/src/types.ts	`84062ecce25bb979e1ec8d2aa6bbc66abbf7bdc8`
packages/snap/src/utils.ts	`6da080f958d479e22386d09ef5c3c02704cb09e4`

The follow-up assessment covered the following files:

File	SHA-1 hash
packages/snap/src/account.ts	`32305836618332b58b57a8e7fa65e11444b216b3`
packages/snap/src/constants.ts	`5637a4484774bd8748b86aeeeab98a19f6ebcb1a`
packages/snap/src/filforwarder.ts	`011922ebd42f81914ce14bee35fa3cca7b1a99a4`

File	SHA-1 hash
packages/snap/src/index.ts	`b19eb8e5ea1dbde19e9185e65535ccd9010c9b00`
packages/snap/src/rpc/configure.ts	`923a405201ba6cb9603203d778e2dc63f8a04880`
packages/snap/src/rpc/export-private-key.ts	`58caa4c7758e8581725e06713d34f122ef01a854`
packages/snap/src/rpc/gas-for-message.ts	`55f414d91e0bf861e53a54117c2223282b4a6f56`
packages/snap/src/rpc/get-account.ts	`78f611f8c6ca391355f89113311ac0ec9a61845a`
packages/snap/src/rpc/get-balance.ts	`73a4654f982f16390afed7f11ce51b6a52cd5cb2`
packages/snap/src/rpc/send-message.ts	`ac5acacdd650bd3f34a6872eb3d7dfec2e7047f9`
packages/snap/src/rpc/sign-message.ts	`8487093bbcd2fbdae3e5a50119729bb78d55b9ce`
packages/snap/src/schemas.ts	`c56fc4148f92b28c9e2264aef1f06b55444f633b`
packages/snap/src/state.ts	`40a874834bcfd2c9512c624883a411c844b47c6a`
packages/snap/src/transaction-insight.ts	`80786c6008b6180ebc11d6212923083df659dfe3`
packages/snap/src/types.ts	`2dda17a8850b6d1cba8d411e8a1c6406109c5040`
packages/snap/src/utils.ts	`121fa94b320d2e1d930e84149b20e8c18530aaf6`

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.