MetaMask/Partner Snaps - StarknetSnap

Date	June 2023
Auditors	Martin Ortner

1 Executive Summary

This report presents the results of our engagement with **ConsenSys** to review their **MetaMask Starknet Snap**.

The review was conducted from **June 26, 2023** to **June 29, 2023**. A total of 4 person-days were spent.

2 Scope

The review focused on the commit hash ec24b0053eae641f49e4095809979c2839758bdf. The list of files in scope can be found in the Appendix.

2.1 Verification Phase

Mitigations were reviewed from **July 18, 2023** to **July 20, 2023** focussing on the commit hash 7231bb7fa4671283b2e7b4cbf5a519d56a57697a.

2.2 Objectives

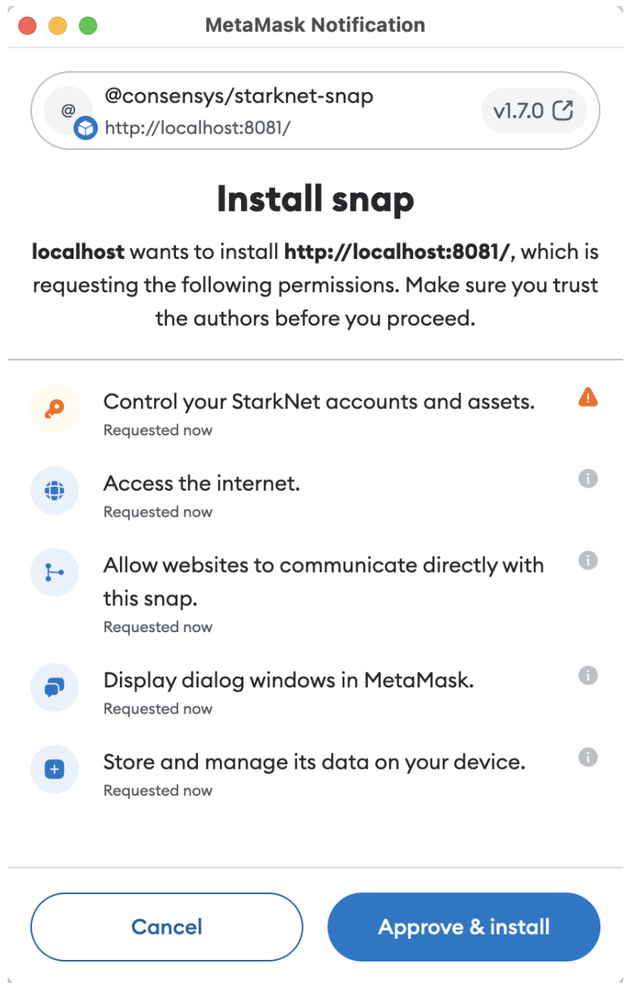
Together with the client, we identified the following priorities for this review:

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 requests access to `coinType:9004` BIP44 Entropy, effectively managing the coins private root key.
- The private key can be displayed to the user within the context of the snap.
- The private key can not be exported to an RPC origin.
- Transactions are signed within the realm of the snap.
- The public key is exposed to connected snaps without additional user confirmation.
- The snap may interact with the following 3rd party service providers via the `fetch()` API:
  - `https://api.coingecko.com`
  - `https://voyager.online`
  - `https://infura.io`
  - `https://starknet.io`
  - and potentially others
- Connected dapps can communicate with the snap via MetaMask snap RPC.

3.1 Capabilities



Permissions

Details

```
👉 2485 (lines of code)
[🚀 == Bundle == ]
    🔺 - bundle (../../dist/bundle.js) does not not exist!
    🔺 - package-lock missing
    🔺 - package.json: invalid license '(Apache-2.0 OR MIT)'
    🔺 - package.json: incomplete package files selection 'undefined'
----%<---- raw permissions
🌐 : https://docs.metamask.io/snaps/reference/rpc-api/#wallet_requestsnaps
{
  snap_dialog: {},
  'endowment:network-access': {},
  snap_getBip44Entropy: [ { coinType: 9004 } ],
  snap_manageState: {},
  'endowment:rpc': { snaps: false, dapps: true }
}
---->%---- raw permissions
🚜 [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/signMessage.ts
       🔶  src/sendTransaction.ts
       🔶  src/recoverAccounts.ts
       🔶  src/index.ts
       🔶  src/extractPrivateKey.ts
       🔶  src/createAccount.ts
🚜 [endowment:network-access]
    🌐 - endowment:network-access - snap can access internet
     ⚠️ - this method may leak information to external api
       🔶  src/utils/starknetUtils.ts
🚜 [snap_getBip44Entropy]
    ✨ - snap_getBip44Entropy - Gets the BIP-44 coin_type key for the coin_type number specified by the method name. See SLIP-44 for the list of availa
       ⚠️ - If you call this method, you receive the user's parent key for the protocol they request. You're managing the user's keys and assets on the
       🔶  src/utils/keyPair.ts
🚜 [snap_manageState]
    🪠 - snap_manageState - snap can store up to 100mb (isolated)
       🔶  src/index.ts
       🔶  src/utils/snapUtils.ts
🚜 [endowment:rpc]
    ❗ - endowment:rpc.dapps - snap can communicate with websites/dapps; check origin for internal api calls!
       🔶  src/index.ts
```

3.2 Dependencies

```
🎄 - Package Depenencies:
     - async-mutex:^0.3.2
     - chai:^4.3.6
     - ethereum-unit-converter:^0.0.17
     - ethers:^5.5.1
     - sinon:^13.0.2
     - sinon-chai:^3.7.0
     - starknet:^4.22.0
     - starknet_v4.6.0:npm:starknet@4.6.0
```

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 RPC `starkNet_sendTransaction` - The User Displayed Message Generated With `getSigningTxnText()` Is Prone to Markdown/Control Chars Injection From `contractCallData`

Major ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by rendering untrusted user input with the copyable UI component, preventing markdown injection. Additionally, the client provided the following statement:

1. restructure dialog ui by using MM copyable field, it can ignore any markdown or tag block
2. validate send transaction calldata has to be able to convert to bigInt

Description

In the code snippet below, `contractCallData` is potentially untrusted and may contain Markdown renderable strings or strings containing Control Characters that break the context of the message displayed to the user. This can lead to misrepresenting the transaction data to be signed, which should be avoided.

packages/starknet-snap/src/utils/snapUtils.ts:L163-L195

```
export function getSigningTxnText(
  state: SnapState,
  contractAddress: string,
  contractFuncName: string,
  contractCallData: string[],
  senderAddress: string,
  maxFee: number.BigNumberish,
  network: Network,
): string {
  // Retrieve the ERC-20 token from snap state for confirmation display purpose
  const token = getErc20Token(state, contractAddress, network.chainId);
  let tokenTransferStr = '';
  if (token && contractFuncName === 'transfer') {
    try {
      let amount = '';
      if ([3, 6, 9, 12, 15, 18].includes(token.decimals)) {
        amount = convert(contractCallData[1], -1 * token.decimals, 'ether');
      } else {
        amount = (Number(contractCallData[1]) * Math.pow(10, -1 * token.decimals)).toFixed(token.decimals);
      }
      tokenTransferStr = `\n\nSender Address: ${senderAddress}\n\nRecipient Address: ${contractCallData[0]}\n\nAmount(${token.sy
    } catch (err) {
      console.error(`getSigningTxnText: error found in amount conversion: ${err}`);
    }
  }
  return (
    `Contract: ${contractAddress}\n\nCall Data: [${contractCallData.join(', ')}]\n\nEstimated Gas Fee(ETH): ${convert(
      maxFee,
      'wei',
      'ether',
    )}\n\nNetwork: ${network.name}` + tokenTransferStr
  );
}
```

packages/starknet-snap/src/sendTransaction.ts:L60-L80

```
const signingTxnText = getSigningTxnText(
  state,
  contractAddress,
  contractFuncName,
  contractCallData,
  senderAddress,
  maxFee,
  network,
);

const response = await wallet.request({
  method: 'snap_dialog',
  params: {
    type: DialogType.Confirmation,
    content: panel([
      heading('Do you want to sign this transaction ?'),
      text(`It will be signed with address: ${senderAddress}`),
      text(signingTxnText),
    ]),
  },
});
```

Please note that we have also reported to the MM Snaps team, that dialogues do not by default hint the origin of the action. We hope this will be addressed in a common way for all snaps in the future,

Recommendation

Validate inputs. Encode data in a safe way to be displayed to the user. Show the original data provided within a pre-text or code-block. Show derived or decoded information (token recipient) as additional information to the user.

4.2 Lax Validation Using `@starknet::validateAndParseAddress` Allows Short Addresses and Does Not Verify Checksums Major ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by wrapping `validateAndParseAddress()` with an implicit length check. Additionally, the client provided the following statement:

1. Add validation on the snap side for address length
2. Checksum will not implement as some users are going to call the Snap directly without going through the dApp

As per the client's decision, checksummed addresses are not enforced.

Description

Address inputs in RPC calls are validated using `@starknet::validateAndParseAddress()`.

packages/starknet-snap/src/getErc20TokenBalance.ts:L19-L28

```
try {
  validateAndParseAddress(requestParamsObj.tokenAddress);
} catch (err) {
  throw new Error(`The given token address is invalid: ${requestParamsObj.tokenAddress}`);
}
try {
  validateAndParseAddress(requestParamsObj.userAddress);
} catch (err) {
  throw new Error(`The given user address is invalid: ${requestParamsObj.userAddress}`);
}
```

While the message validates the general structure for valid addresses, it does not strictly enforce address length and may silently add padding to the inputs before validation. This can be problematic as it may hide user input errors when a user provides an address that is too short and silently gets left-padded with zeroes. This may unintentionally cause a user to request action on the wrong address without them recognizing it.

../src/utils/address.ts:L14-L24

```
export function validateAndParseAddress(address: BigNumberish): string {
  assertInRange(address, ZERO, MASK_251, 'Starknet Address');

  const result = addAddressPadding(address);

  if (!result.match(/^(0x)?[0-9a-fA-F]{64}$/)) {
    throw new Error('Invalid Address Format');
  }

  return result;
}
```

```
export function validateAndParseAddress(address: BigNumberish): string {
  assertInRange(address, ZERO, MASK_251, 'Starknet Address');

  const result = addAddressPadding(address);

  if (!result.match(/^(0x)?[0-9a-fA-F]{64}$/)) {
    throw new Error('Invalid Address Format');
  }

  return result;
}
```

Recommendation

The exposed Snap API should strictly validate inputs. User input must be provided in a safe canonical form (exact address length, checksum) by the dapp.

4.3 RPC `starkNet_signMessage` - Fails to Display the User Account That Is Used for Signing the Message Major ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by displaying the signing accounts address with the dialog. All user-provided fields are copyable, preventing any markdown injection. Additionally, the client provided the following statement:

1. add signer address add bottom of the dialog

We want to note that the origin of the RPC call is not visible in the dialog. However, we recommend addressing this with the MM Snap SDK by generically showing the origin of MM popups with the dialog.

Description

The signing request dialogue does not display the user account that is being used to sign the message. A malicious dapp may pretend to sign a message with one account while issuing an RPC call for a different account.

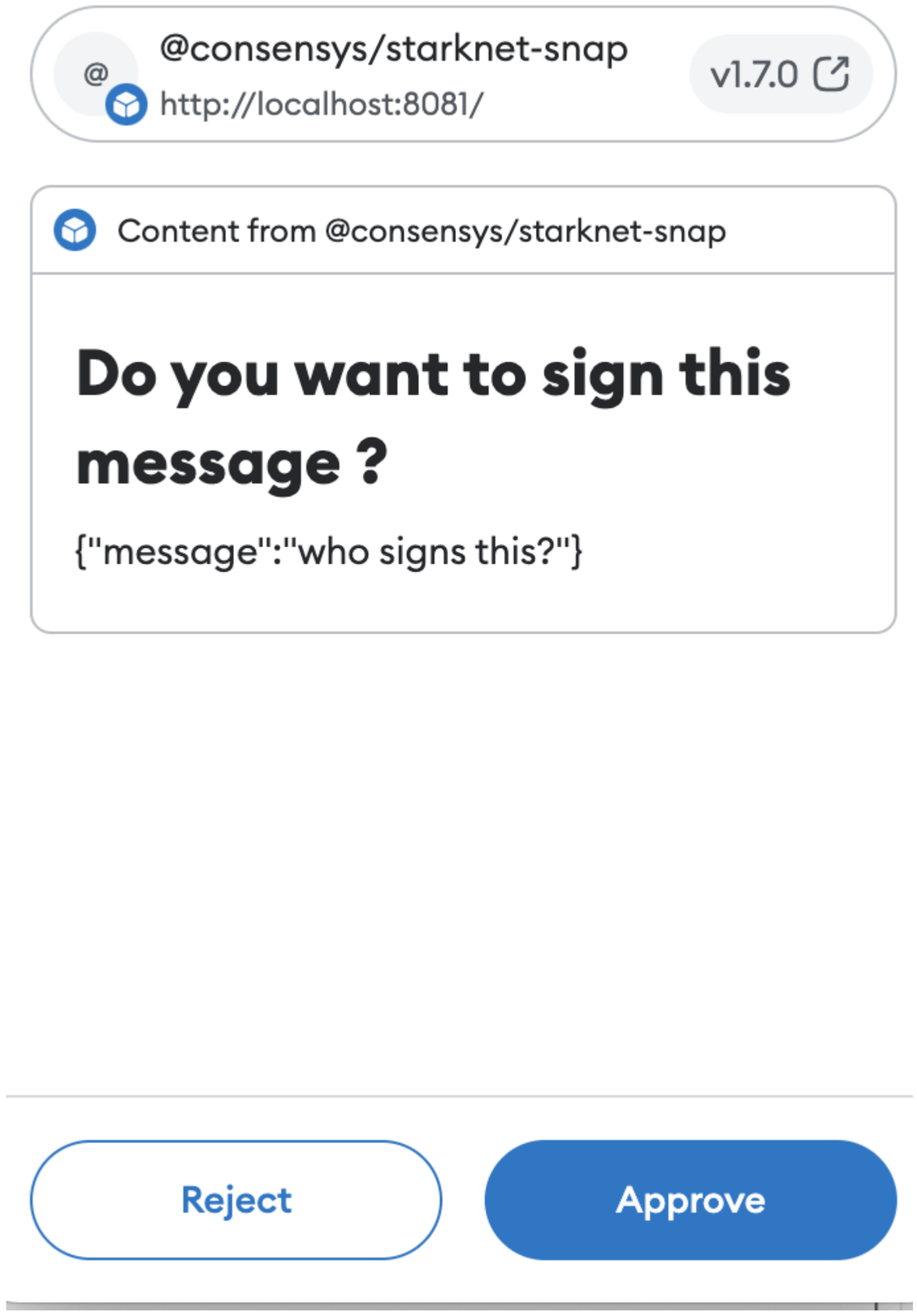
Note that StarkNet signing requests should implement similar security measures to how MetaMask signing requests work. Being fully transparent on “who signs what”, also displaying the origin of the request. This is especially important on multi-dapp snaps to avoid users being tricked into signing transactions they did not intend to sign (wrong signer).

packages/starknet-snap/src/signMessage.ts:L34-L42

```
const response = await wallet.request({
  method: 'snap_dialog',
  params: {
    type: DialogType.Confirmation,
    content: panel([heading('Do you want to sign this message ?'), text(JSON.stringify(typedDataMessage))]),
  },
});
if (!response) return false;
```

Examples

UI does not show the signing accounts address. Hence, the user cannot be sure what account is used to sign the message.



Recommendation

Show what account is requested to sign a message. Display the origin of the RPC call.

4.4 RPC `starkNet_signMessage` - Inconsistency When Previewing the Signed Message (Markdown Injection) Major ✓ Fixed

Resolution
Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by rendering user-provided information with the copyable UI component. Additionally, the client provided the following statement: 1. restructure dialog ui by using MM copyable field, it can ignore any markdown or tag block

Description

The snap displays an dialogue to the user requesting them to confirm that they want to sign a message when a dapp performs a request to `starkNet_signMessage`. However, the MetaMask Snaps UI `text()` component will render Markdown. This means that the

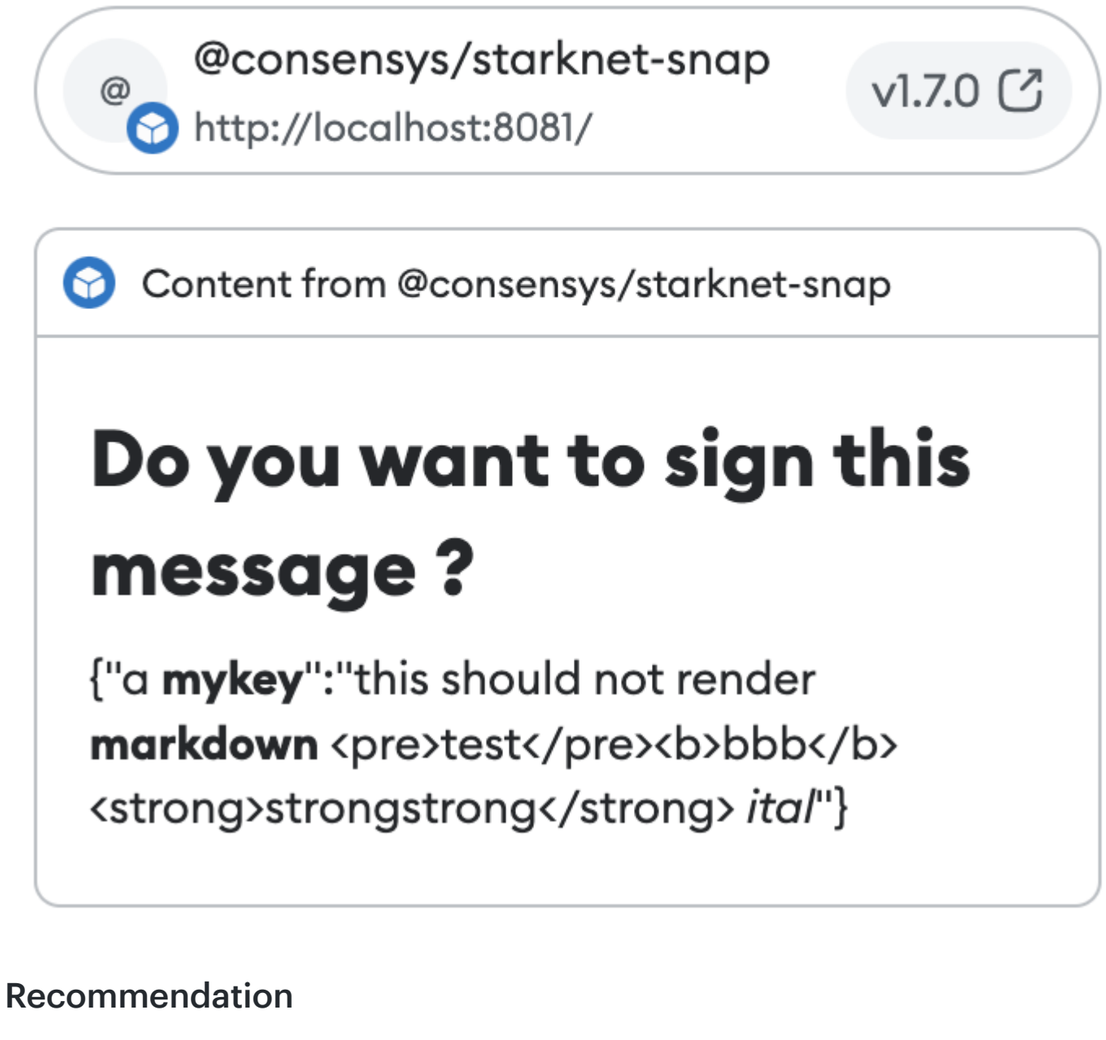
message-to-be-signed displayed to the user for approval will be inaccurate if it contains Markdown renderable text.

packages/starknet-snap/src/signMessage.ts:L35-L41

```
const response = await wallet.request({
  method: 'snap_dialog',
  params: {
    type: DialogType.Confirmation,
    content: panel([heading('Do you want to sign this message ?'), text(JSON.stringify(typedDataMessage))]),
  },
});
```

Examples

```
{"a **mykey**":"this should not render **markdown** <pre>test</pre><b>bbb</b><strong>strongstrong</strong>[visit oststrom](https
```



Recommendation

Render signed message contents in a code block or preformatted text blocks.

Note: we've also reported this to the MetaMask Snaps team to provide further guidance.

4.5 UI/AlertView - Unnecessary Use of `dangerouslySetInnerHTML` Medium ✓ Fixed

Resolution
Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by not using `dangerouslySetInnerHTML`. Additionally, the client provided the following statement: 1. Remove dangerouslySetInnerHTML from UI

Description

`AlertView` is populated by setting `innerHTML` instead of the component's value, which would be auto-escaped. This only makes sense if the component is supposed to render HTML. However, the component is never used with HTML as input, and the attribute name `text` is misleading.

packages/wallet-ui/src/components/ui/atom/Alert/Alert.view.tsx:L11-L36

```
export function AlertView({ text, variant, ...otherProps }: Props) {
  const paragraph = useRef<HTMLParagraphElement | null>(null);
  const [isMultiline, setIsMultiline] = useState(false);
  useEffect(() => {
    if (paragraph.current) {
      const height = paragraph.current.offsetHeight;
      setIsMultiline(height > 20);
    }
  }, []);
  return (
    <Wrapper isMultiline={isMultiline} variant={variant} {...otherProps}>
      <>
        {variant === VariantOptions.SUCCESS && <LeftIcon icon={['fas', 'check-circle']} />}
        {variant === VariantOptions.INFO && <LeftIcon icon={['fas', 'info-circle']} color={theme.palette.info.dark} />}
        {variant === VariantOptions.ERROR && (
          <LeftIcon icon={['fas', 'exclamation-circle']} color={theme.palette.error.main} />
        )}
        {variant === VariantOptions.WARNING && (
          <LeftIcon icon={['fas', 'exclamation-triangle']} color={theme.palette.warning.main} />
        )}
        <Parag ref={paragraph} color={variant} dangerouslySetInnerHTML={{ __html: text }} />
      </>
    </Wrapper>
  );
}
```

packages/wallet-ui/src/components/ui/organism/NoFlaskModal/NoFlaskModal.view.tsx:L4-L25

```
export const NoFlaskModalView = () => {
  return (
    <Wrapper>
      <StarknetLogo />
      <Title>You don't have the MetaMask Flask extension</Title>
      <DescriptionCentered>
        You need to install MetaMask Flask extension in order to use the StarkNet Snap.
        <br />
        <br />
        <AlertView
          text="Please make sure that the regular MetaMask extension is disabled or use a different browser profile"
          variant="warning"
        />
      </DescriptionCentered>
      <a href="https://metamask.io/flask" target="_blank" rel="noreferrer noopener">
        <ConnectButton customIconLeft={<FlaskIcon />} onClick={() => {}}>
          Download MetaMask Flask
        </ConnectButton>
      </a>
    </Wrapper>
  );
};
```

Setting HTML from code is risky because it's easy to inadvertently expose users to a cross-site scripting (XSS) attack.

Recommendation

Do not use `dangerouslySetInnerHTML` unless there is a specific requirement that passed in HTML be rendered. If so, rename the attribute name to `html` instead of `text` to set clear expectations regarding how the input is treated. Nevertheless, since the component is not used with HTML input, we recommend removing `dangerouslySetInnerHTML` altogether.

4.6 RPC `starkNet_addErc20Token` - Should Ask for User Confirmation Medium ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by requesting user confirmation for adding new ERC20 Tokens. Additionally, the client provided the following statement:

1. Adding confirm dialog with MM copyable field, it can ignore any markdown or tag block
2. Disable loading frame when user reject the add ec220 token request on UI

Description

The RPC method upserts ERC20 tokens received via RPC without asking the user for confirmation. This would allow a connected dapp to insert/change ERC20 token information anytime. This can even be more problematic when multiple dapps are connected to the StarkNet-Snap (race conditions).

packages/starknet-snap/src/addErc20Token.ts:L30-L47

```
  validateAddErc20TokenParams(requestParamsObj, network);

  const erc20Token: Erc20Token = {
    address: tokenAddress,
    name: tokenName,
    symbol: tokenSymbol,
    decimals: tokenDecimals,
    chainId: network.chainId,
  };

  await upsertErc20Token(erc20Token, wallet, saveMutex);

  console.log(`addErc20Token:\nerc20Token: ${JSON.stringify(erc20Token)}`);
  return erc20Token;
} catch (err) {
  console.error(`Problem found: ${err}`);
  throw err;
}
```

Recommendation

Ask the user for confirmation when changing the snaps state.

4.7 `getKeysFromAddress` - Possible Unchecked Null Dereference When Looking Up Private Key

Medium ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by throwing an exception on error. Additionally, the client provided the following statement:

1. instead of return null, raise err in getKeysFromAddress, caller will catch the exception

Description

`getKeysFromAddress()` may return `null` if an invalid address was provided but most callers of the function do not check for the `null` condition and blindly dereference or unpack the return value causing an exception.

packages/starknet-snap/src/utils/starknetUtils.ts:L453-L455

```
  }
  return null;
};
```

Examples

packages/starknet-snap/src/signMessage.ts:L44-L46

```
const { privateKey: signerPrivateKey } = await getKeysFromAddress(keyDeriver, network, state, signerAddress);
const signerKeyPair = getKeyPairFromPrivateKey(signerPrivateKey);
const typedDataSignature = getTypedDataMessageSignature(signerKeyPair, typedDataMessage, signerAddress);
```

packages/starknet-snap/src/extractPrivateKey.ts:L37

```
const { privateKey: userPrivateKey } = await getKeysFromAddress(keyDeriver, network, state, userAddress);
```

packages/starknet-snap/src/extractPublicKey.ts:L31-L32

```
const { publicKey } = await getKeysFromAddress(keyDeriver, network, state, userAddress);
userPublicKey = publicKey;
```

packages/starknet-snap/src/sendTransaction.ts:L48-L52

```
const {
  privateKey: senderPrivateKey,
  publicKey,
  addressIndex,
} = await getKeysFromAddress(keyDeriver, network, state, senderAddress);
```

packages/starknet-snap/src/signMessage.ts:L44-L45

```
const { privateKey: signerPrivateKey } = await getKeysFromAddress(keyDeriver, network, state, signerAddress);
const signerKeyPair = getKeyPairFromPrivateKey(signerPrivateKey);
```

packages/starknet-snap/src/verifySignedMessage.ts:L38

```
const { privateKey: signerPrivateKey } = await getKeysFromAddress(keyDeriver, network, state, verifySignerAddress);
```

packages/starknet-snap/src/estimateFee.ts:L48-L53

```
const { privateKey: senderPrivateKey, publicKey } = await getKeysFromAddress(
  keyDeriver,
  network,
  state,
  senderAddress,
);
```

Recommendation

Explicitly check for the `null` or `{}` case. Consider returning `{}` to not allow unpacking followed by an explicit null check.

4.8 RPC `starkNet_getStoredTransactions` - Lax or Missing Input Validation Minor Won't Fix

Resolution
Won't fix. The client provided the following statement: not fix, minor impact We want to note that strict input validation should be performed on all untrusted inputs for read/write and read-only methods. Just because the method is read-only now does not necessarily mean it will stay that way. Leaving untrusted inputs unchecked may lead to more severe security vulnerabilities with a growing codebase in the future.

Description

Potentially untrusted inputs, e.g. addresses received via RPC calls, are not always checked to conform to the StarkNet address format. For example, `requestParamsObj.senderAddress` is never checked to be a valid StarkNet address.

packages/starknet-snap/src/getStoredTransactions.ts:L18-L26

```
const transactions = getTransactions(
  state,
  network.chainId,
  requestParamsObj.senderAddress,
  requestParamsObj.contractAddress,
  requestParamsObj.txnType,
  undefined,
  minTimeStamp,
);
```

Recommendation

This method is read-only, and therefore, severity is estimated as Minor. However, it is always suggested to perform strict input validation on all user-provided inputs for read-only and read-write methods.

4.9 Disable Debug Log for Production Build Minor ✓ Fixed

Resolution
Addressed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a by introducing a configurable logger. Additionally, the client provided the following statement: 1. add custom logger to replace console.log, and log message base on debug level, when debug level is off, it will not log anything 2. update production CICD pipeline to build project with debug level = off/disabled There're still some instances of `console.log()`. However, internal state or full requests are not logged anymore. We would still recommend replacing the remaining `console.log` calls (e.g. the one in addERC20Token).

Description

Throughout the codebase, there are various places where debug log output is being printed to the console. This should be avoided for production builds.

Examples

packages/starknet-snap/src/index.ts:L45-L46

```
// Switch statement for methods not requiring state to speed things up a bit
console.log(origin, request);
```

packages/starknet-snap/src/index.ts:L91-L92

```

console.log(`${request.method}:\nrequestParams: ${JSON.stringify(requestParams)}`);
```

packages/starknet-snap/src/index.ts:L103

```
console.log(`Snap State:\n${JSON.stringify(state, null, 2)}`);
```

Recommendation

Remove the debug output or create a custom log method that allows to enable/disable logging to console.

4.10 package.json - Dependecy Mixup Minor ✓ Fixed

Resolution

Fixed with Consensys/starknet-snap@7231bb7fa4671283b2e7b4cbf5a519d56a57697a as per recommendation. Additionally, the client provided the following statement:

> Move development dependencies to package.json::devDependencies

Description

The following dependencies are only used for testing or development purposes and should therefore be listed as `devDependencies` in `package.json`, otherwise they may be installed for production builds, too.

- https://sinonjs.org/
- https://www.chaijs.com/

packages/starknet-snap/package.json:L50

```
"chai": "^4.3.6",
```

packages/starknet-snap/package.json:L53-L54

```
"sinon": "^13.0.2",
"sinon-chai": "^3.7.0",
```

Recommendation

Move development dependencies to `package.json::devDependencies`.

4.11 package.json - Invalid License Minor ✓ Invalid

Resolution

Invalid. Legal clarified that it is perfectly fine to allow MIT+Apache2. Additionally, client provided the following statement:

> not fix, choose to stick with dual license

Description

The license field in `package.json` is invalid.

packages/starknet-snap/package.json:L4

```
"license": "(Apache-2.0 OR MIT)",
```

Recommendation

Update the license field.

4.12 RPC `starkNet_extractPrivateKey` - Should Be Renamed to `starkNet_displayPrivateKey` Won't Fix

Resolution

Won’t Fix. The client provided the following statement:

> not fix, the extractPrivateKey is not for display purpose

We want to note that we still encourage changing the method name and return value to explicitly return `null` in the RPC handler for the sake of good secure coding practices discouraging future devs to return implementing key extraction RPC endpoints that may expose wallet credentials to a linked dapp.

Description

It is recommended to rename `starkNet_extractPrivateKey` to `starkNet_displayPrivateKey` as this more accurately describes what the RPC method is doing.

Also, the way the method handler is implemented makes it appear as if it returns the private key to the RPC origin while the submethod returns `null`. Consider changing this to an explicit empty `return` to clearly mark in the outer call that no private key is exposed to the caller. Not to confuse this with how `starkNet_extractPublicKey` works which actually returns the pubkey to the RPC caller.

packages/starknet-snap/src/index.ts:L123-L127

```
case 'starkNet_extractPrivateKey':
  apiParams.keyDeriver = await getAddressKeyDeriver(snap);
  return extractPrivateKey(apiParams);
```

4.13 UI/hooks - `detectEthereumProvider()` Should Require `mustBeMetaMask` Won't Fix

Resolution
Won’t Fix. The client provided the following statement: not fix, minor impact

Description

MetaMask Snaps require a Metamask provider. However, `detectEthereumProvider()` does not explicitly require a MetaMask provider and would continue if the alternative provider contains the substring `flask` in their signature.

packages/wallet-ui/src/hooks/useHasMetamaskFlask.ts:L7-L16

```
const detectMetamaskFlask = async () => {
  try {
    const provider = (await detectEthereumProvider({
      mustBeMetaMask: false,
      silent: true,
    })) as any | undefined;
    const isFlask = (await provider?.request({ method: 'web3_clientVersion' }))?.includes('flask');
    if (provider && isFlask) {
      return true;
    }
```

Consider requiring `mustBeMetaMask = true` to enforce that the injected provider is indeed MetaMask. This will also work with MetaMask Flask as shown here:

```
⇒ window.ethereum.isMetaMask
true
⇒ await window.ethereum.request({ method: 'web3_clientVersion' })
'MetaMask/v10.32.0-flask.0'
```

4.14 RPC `starkNet_addNetwork` - Not Implemented, No User Confirmation Won't Fix

Resolution
Won’t Fix. The client provided the following statement: not fix, minor impact

Description

It was observed that the RPC method `starkNet_addNetwork` is not implemented.

In case this method is to be exposed to dapps, we recommended to follow the advise given in issue 4.6 to ask for user confirmation when adjusting the snaps configuration state.

Appendix 1 - Files in Scope

#	Total	Code	Comment	ToDo	Name	Sha1
1	48	41			src/addErc20Token.ts	aee1b126f61f99befc73974f6eb1568a44036679
2	28	24			src/addNetwork.ts	17cb8e1ecf264dadc8d011c25a92090e546166a8
3	163	149	1		src/createAccount.ts	6b0909c362674c349e6c6b9769ee6cdea5b01713

#	Total	Code	Comment	ToDo	Name	Sha1
4	54	48			src/estimateAccountDeployFee.ts	5f30f28d0490edab6a8382338d5fe2d339eaac6d
5	127	111	3		src/estimateFee.ts	0cfdbfbef81cc5c6cce72151aecf34a803cfa795
6	53	46			src/extractPrivateKey.ts	98c94715bc17cb4bbadd8c38e08bc08c719e27c3
7	44	37			src/extractPublicKey.ts	688c5c52bab89b0b22a69f6cfce89bfa59cbd5b3
8	46	37	1		src/getErc20TokenBalance.ts	37d3734bfbfd886db497f11e98c95359679cb4c3
9	18	15			src/getStoredErc20Tokens.ts	8aa92a309746084f0185b6e476b5f38cd472e890
10	16	13			src/getStoredNetworks.ts	b15c50d24a57b711c39977a11625c0e089c07783
11	34	30	1		src/getStoredTransactions.ts	d8d7053e9b49cbc71e18658632a66cde4bb10706
12	22	17	1		src/getStoredUserAccounts.ts	15b946222bee92227f89907f7794a13e02d9807d
13	21	17			src/getTransactionStatus.ts	c29cdf498cf328f36f49ba50c73e0c471137963c
14	141	118	10		src/getTransactions.ts	ad6a70311ac385268de7aad69b46d5e86a72b33e
15	44	37			src/getValue.ts	f3035b134c504fb0279ccd2ae8a7e70b5edb3aa4
16	186	154	3		src/index.ts	c0da950dd27b239a93bc48ace593fbda06604085
17	97	82			src/recoverAccounts.ts	b213e16d99cdb178550c9deba28f523c792fac01
18	142	126	2		src/sendTransaction.ts	30439541fd07c87a76a1f254d2ddf4c430db1a3f
19	54	46			src/signMessage.ts	37fbf18730c4a7f93446ad0a0ea638f8f02ec836
20	143	118	3		src/types/snapApi.ts	a94956047e420b64926df78c313031cbb9982e0c
21	71	63	15		src/types/snapState.ts	21eacb5fb3c257e1096f95f0b055a208614244cb
22	162	141	23		src/utils/constants.ts	f4a91bc93aca2f33628efbd9cfb37567fe5bc034
23	57	46	4		src/utils/keyPair.ts	b204d7157b3f7ba10889901b6441b68fde4eba14
24	531	473	2		src/utils/snapUtils.ts	6b371d0d7af027bcc9a83fcd42f1756d25f13853
25	508	450	24		src/utils/starknetUtils.ts	78c9f02da21bd3137aa220290aeff4ddc6a80fb7
26	54	46			src/verifySignedMessage.ts	7259f7b8d33b2e05acbbc23d9986d58d3e52a548
=====	=====	=====	=====	=====		
Σ	2864	2485	93	0		

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 bugfree 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.

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.

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.

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.