Push Protocol Snap for MetaMask

Date	July 2023
Auditors	Martin Ortner

1 Executive Summary

This report presents the results of our engagement with **PushProtocol** to review **Snap v1**, a MetaMask Snap for delivering channel notifications and chat notifications within metamask wallet.

The review was conducted from **July 5, 2023** to **July 6, 2023**. A total of 2 person-days were spent.

2 Scope

The review focused on the commit hash c1636586dee1e43cd447f9c4ac03b0d776224f9c. The list of files in scope can be found in the Appendix.

The Technical Specification can be found here.

2.1 Verification Phase

Mitigations were reviewed from **July 11, 2023** to **July 13, 2023** focussing on the commit hash 1a6a32ef760088ca59f73e555f41b5b5d871f761.

Update 1:

The client provided commit hash 1a6a32ef760088ca59f73e555f41b5b5d871f761 for review addressing outstanding issues on **July 14, 2023**.

Update 2:

The client provided commit hash b40e141243c77bfd7ec109408b326607b19314c8 for review addressing outstanding issues on **July 17, 2023**.

2.2 Objectives

Together with the **Push Protocol** 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 Document Change Log

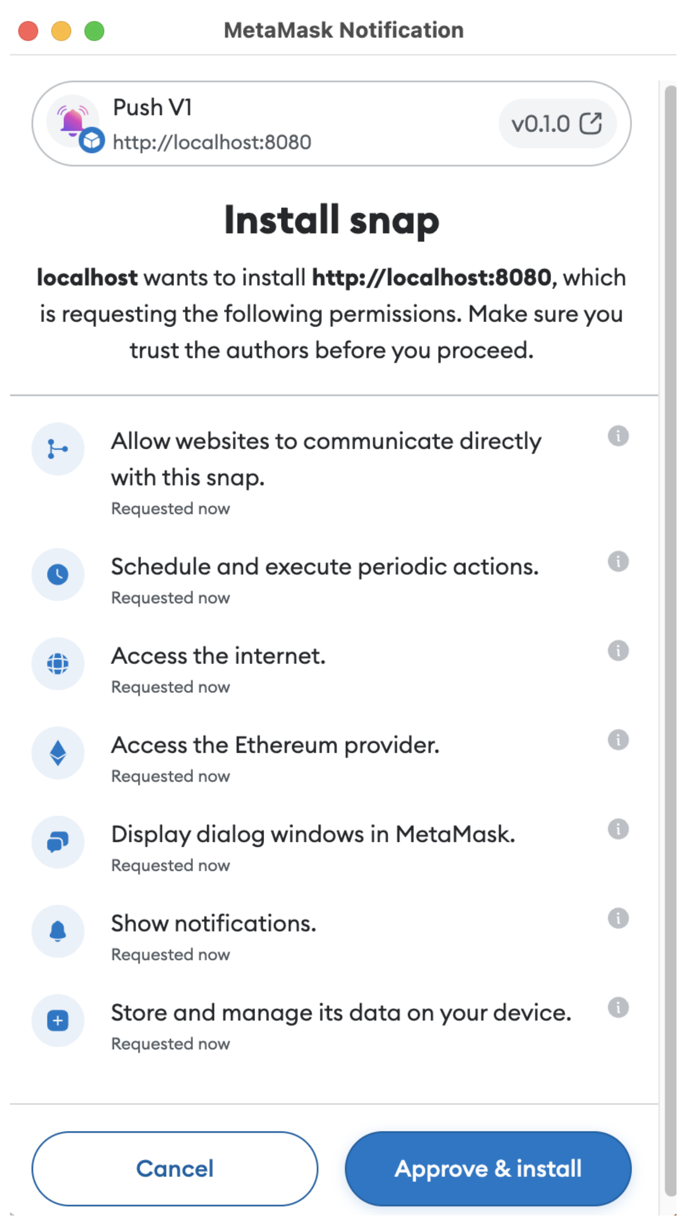
Version	Date	Description
1.0	2023-07-06	Initial report
1.1	2023-07-12	Mitigations review
1.2	2023-07-14	Mitigations review, update 1
1.3	2023-07-17	Mitigations review, update 2

4 Snap Outline

- The snap stores addresses to be monitore and config settings in `snap_manageState`.
- The snap may interact with the following 3rd party service providers via the `fetch()` API:
  - `https://backend-prod.epns.io/apis/v1/users/eip155:5:${address}/feeds`
- Connected dapps can communicate with the snap via MetaMask snap RPC.
- The Snap registers a cronjob that fires every minute, fetching new notifications.

Note: Notification messages for all addresses are public. Anyone can subscribe to notifications sent to any address. The back-end API does not authenticate the requested address (signature). The wallet may require proof of ownership but this is only implemented in the front-end (dapp) and can easily be bypassed.

4.1 Capabilities


MetaMask Notification

Push V1
http://localhost:8080
v0.1.0

Install snap

localhost wants to install **http://localhost:8080**, which is requesting the following permissions. Make sure you trust the authors before you proceed.

- Allow websites to communicate directly with this snap.
  Requested now
- Schedule and execute periodic actions.
  Requested now
- Access the internet.
  Requested now
- Access the Ethereum provider.
  Requested now
- Display dialog windows in MetaMask.
  Requested now
- Show notifications.
  Requested now
- Store and manage its data on your device.
  Requested now

Permissions

Details

```
👉 249 (lines of code)
[🚀 == Bundle == ]
    🔺 Error: Missing file "images/icon.svg".
    🔺 - package-lock missing
    🔺 - package.json: invalid license '(MIT-0 OR Apache-2.0)'
----%<---- raw permissions
🌐 : https://docs.metamask.io/snaps/reference/rpc-api/#wallet_requestsnaps
{
  snap_dialog: {},
  'endowment:rpc': { dapps: true, snaps: false },
  snap_notify: {},
  'endowment:cronjob': { jobs: [ [Object] ] },
  'endowment:network-access': {},
  snap_manageState: {},
  'endowment:ethereum-provider': {}
}
---->%---- 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/index.ts
      🔶  src/utils/fetchAddress.ts
🚜 [endowment:rpc]
    ❗ - endowment:rpc.dapps - snap can communicate with websites/dapps; check origin for internal api calls!
      🔶  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
🚜 [endowment:cronjob]
    ❗ - endowment:cronjob.fireCronjob - fires every minute! (* * * * *)
      🔶  src/index.ts
🚜 [endowment:network-access]
    🌐 - endowment:network-access - snap can access internet
     ⚠️ - this method may leak information to external api
      🔶  src/index.ts
      🔶  src/utils/fetchnotifs.ts
🚜 [snap_manageState]
    📦 - snap_manageState - snap can store up to 100mb (isolated)
      🔶  src/index.ts
      🔶  src/utils/toggleHelper.ts
      🔶  src/utils/fetchAddress.ts
🚜 [endowment:ethereum-provider]
    🔺 - endowment:ethereum-provider - snap can access ethereum API
    ⚠️ - check if the **snap code** (not site) actually accesses the global object 'window.ethereum'
      🔺 - superfluous permission: no reference to 'window.ethereum.*'!
```

4.2 Dependencies

```
🌲 - Package Depenencies:
    - ethers:5.7.2
```

5 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.

5.1 Superfluous Permission `endowment:ethereum-provider` Major ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by removing the ethereum provider permission from the manifest.

Description

The snap requests permission `endowment:ethereum-provider` but `window.ethereum` is never accessed from within the snap's context.

snap/snap.manifest.json:L39

```
"endowment:ethereum-provider": {}
```

Recommendation

Remove superfluous permissions.

5.2 A Trusted Website Can Add Any Address to the Snaps Address Storage; No Control Over Added Addresses; Confirmation Is a Notification Major ✓ Fixed

Resolution

partially addressed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by only allowing trusted origins to interact with the snap.

Update: user confirmation for address management (add/remove current account) added with ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829

Description

Trusted websites can add addresses to the list of addresses the user wants to receive notifications for. However, the user has no control over the addresses, and even though the code suggests that the snap user must confirm new address addition, this confirmation is merely a notification that the address has been added.

The lack of address management may lead to a self-DoS when too many addresses are added to the extension.

```
await window.ethereum?.request({
      method: "wallet_invokeSnap",
      params: {
        snapId: "local:http://localhost:8080",
        request: { method: 'hello', params: { address: "\nhi\nho" } },
      }})
```

push-snap-site/components/buttons/ConfirmButton.tsx:L6-L35

```
export default function ConfirmButton() {
  const { address, isConnecting, isDisconnected } = useAccount();
  const defaultSnapOrigin = `local:http://localhost:8080`;

  const sendHello = async (address: string) => {
    await window.ethereum?.request({
      method: "wallet_invokeSnap",
      params: {
        snapId: defaultSnapOrigin,
        request: { method: 'hello', params: { address: address } },
      },
    });
  };

  const { data, isError, isLoading, isSuccess, signMessage } = useSignMessage({
    message:
      `Confirm your Address ${address}, \n this will be added to MetaMask for sending notifications`,
  });

  function sleep(ms: number) {
    return new Promise((resolve) => setTimeout(resolve, ms));
  }

  const confirmAddition=async()=>{
    signMessage();
    if(isSuccess){
      await sleep(5000);
      await sendHello(String(address));
    }
  }
```

The same is true for configuration settings. Any connected dap may set `togglepopup`. This may be problematic in multi-dapp scenarios where multiple dapps request to set `togglepopup`.

Recommendation

Note that dapps are not necessarily completely trusted. They can be modified, or malicious behavior may be added later by the dapp deployer (unless used locally or via IPFS). Therefore, the snap should always notify the wallet owner of important state changes and allow them to reject them or, in this case, manage addresses that've been added previously.

Consider checking the origin in `onRPC` if this dapp is only meant to be called from a specific dapp address. Otherwise, any connected dapp may change configuration settings.

5.3 Lax Input Validation, Control Char, URI, and Markdown Injection Major ✓ Fixed

Resolution

addressed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 validating the address with `ethers.utils.isAddress`.

Update 1:

- Markdown rendering of newlines fixed with: ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829
- **Major:** Markdown Injection in Confirmation Dialogue re-introduced with ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829

Update 2:

Markdown Injection in Confirmation Dialogue fixed with ethereum-push-notification-service/push-protocol-snaps@b40e141243c77bfd7ec109408b326607b19314c8

Description

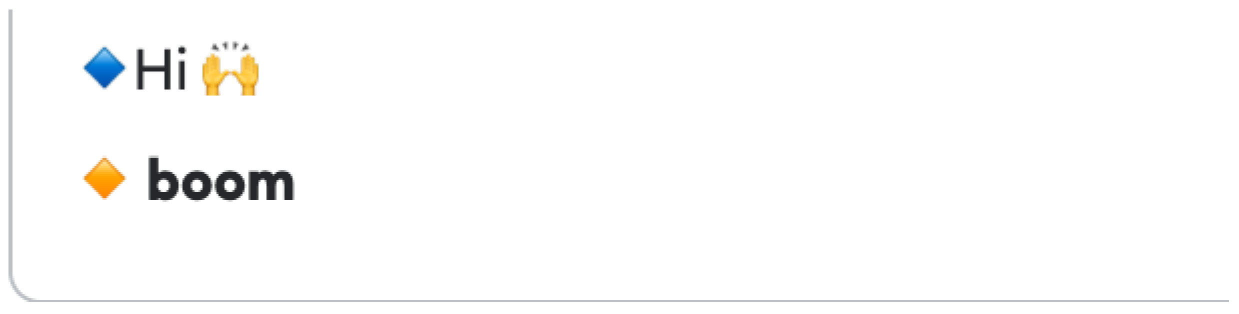
There is no input validation on the `address` to be added. The input may be an ethereum address but can be anything, potentially breaking security assumptions in the code and leading to unwanted side effects.

- `request.params` may be `null`, and
- `request.params.address` may not be an ethereum address.

snap/src/index.ts:L18

```
await addAddress(request.params.address || "0x0");
```

Example


🔷Hi 🙌

🔶 **boom**

```
await window.ethereum?.request({
      method: "wallet_invokeSnap",
      params: {
        snapId: "local:http://localhost:8080",
        request: { method: 'hello', params: { address: "Hi 🙌\n\n 🔶 **boom**" } },
      }})
```

- URI injection if address contains `?#/`

snap/src/utils/fetchnotifs.ts:L3-L13

```
export const getNotifications=async(address:string)=>{
    const url = `https://backend-prod.epns.io/apis/v1/users/eip155:5:${address}/feeds`;
    const response = await fetch(url, {
      method: 'get',
      headers: {
        'Content-Type': 'application/json',
      },
    });
    const data = await response.json();
    return data;
  }
```

- Injection in notifications

snap/src/utils/popupHelper.ts:L3-L12

```
export const popupHelper = (notifs: String[]) => {
  let msg = [];
  if (notifs.length > 0) {
    notifs.forEach((notif) => {
      let str = `\n🔔` + notif + "\n";
      msg.push(str);
    });
  }
  return msg;
};
```

- Markdown injection

snap/src/utils/fetchAddress.ts:L45-L52

```
const data = persistedData.addresses;
const popup = persistedData.popuptoggle;
let msg='';
for(let i = 0; i < data!.length; i++){
    msg = msg + '🔷' + data![i] + '\n';
}
return snap.request({
    method: 'snap_dialog',
```

Also, note that the currently rendered markdown that lists addresses appears wrong, as markdown newlines require `\n\n` instead of `\n`.

Recommendation

Strictly validate inputs from external origins. Ensure that the provided address is a valid ethereum address. Optionally check the addresses checksum to detect typos. Ensure that inputs may not lead to renderable markdown. Fix the rendered list of addresses to properly display as a newline'd list. Ensure untrusted inputs cannot inject context-sensitive information into fetch urls.

5.4 `persistedData` Race Where `snap_manageState.get` Returns `null` Major ✓ Fixed

Resolution

addressed with ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829 by introducing a wrapper function that ensures that snapstate returns sane defaults. This function is not used everywhere, but in places where it is not, custom checks are employed.

Description

Metamask Error:

```
Oops! Something went wrong.
Snap Error: 'Cannot read properties of null (reading 'addresses')'. Error Code: '-32603'
```

`snap.request(, {method: 'snap_manageState', params: {operation: 'get'}})` may return `null`. Snap state is only initialized on rpc request method `hello` via `addAddress()`.

This is the only method that checks if the retrieved state is `null`:

snap/src/utils/fetchAddress.ts:L5-L20

```
export const addAddress = async (address:string) => {

    const persistedData = await snap.request({
        method: 'snap_manageState',
        params: { operation: 'get' },
    });

    if(persistedData == null){
        const data = {
            addresses: [address],
            popuptoggle: 0,
        };
        await snap.request({
            method: 'snap_manageState',
            params: { operation: 'update', newState:data },
        });
```

snap/src/index.ts:L12-L21

```
export const onRpcRequest: OnRpcRequestHandler = async ({
  origin,
  request,
}) => {
  switch (request.method) {
    case "hello": {
      await addAddress(request.params.address || "0x0");
      await confirmAddress();
      break;
    }
```

If the state was never initialized or there was a race where `rpc-hello()` was not called first, then the snap may run into a null deref exception (here `rpc-togglepopup`):

snap/src/utils/toggleHelper.ts:L2-L12

```
let persistedData = await snap.request({
    method: 'snap_manageState',
    params: { operation: 'get' },
});

let popuptoggle = notifcount;

const data = {
    addresses: persistedData.addresses,
    popuptoggle: popuptoggle,
};
```

Recommendation

Wrap `snap_manageState` with a function that always falls back to safe defaults if the snap state was never set. This also obsoleted the future need to check if `persistedData` is `null` as the new method ensures safe non-null defaults.

This should also silence some of the type errors reported by tslint that warn that attributes of `persistentdata` are read while it might be null (see issue 5.6).

5.5 User Flow - Request to Sign Message Does Not Provide Security Guarantee Medium ✓ Fixed

Resolution

obsolete, removed with ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829.

Description

A connected dapp can add any address to the snap via the RPC method `hello`. There is no added security by requesting the user to sign with their address as the backend API gives access to any address notification (they are not private) and the dapps request is a front-end-only solution. A user may add any other address by creating their dapp which allows custom addresses.

In light of this, the front-end (dapp) security check requiring the user to prove that they are in possession of the private key appears not to add any security guarantees to the snap. Instead, the snap may want to enumerate wallet account addresses internally instead and remove the `hello` API altogether, or, allow any address to be added without requiring a proof of ownership of an address.

Examples

push-snap-site/components/buttons/ConfirmButton.tsx:L20-L23

```
const { data, isError, isLoading, isSuccess, signMessage } = useSignMessage({
  message:
    `Confirm your Address ${address}, \n this will be added to MetaMask for sending notifications`,
});
```

Recommendation

Remove the signature check, and add linked accounts from within the snaps context. Be transparent that notification texts are not private, and anyone can subscribe to the back-end API. If notifications are private to the recipient, we suggest encrypting them for the target account and adding logic in the snap to allow the recipient to decrypt them within the context of the snap.

5.6 TypeScript Errors Medium ✓ Fixed

Resolution

partially addressed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761:

- `toggleHelper` not addressed.
- `popupHelper` addressed as per recommendation.
- `fetchAllAddrNotifs` fixed by forcing `fetchAddress()` to return empty array instead.
- `persistedData` partially addressed. might still null-deref at `persistedData.addresses` in `index.ts`

Update: `toggleHelper` and `persistedData` addressed with the snap data check wrapper function in ethereum-push-notification-service/push-protocol-snaps@7ee018947303014e8c14e9413a5edd9fd29f9829

Description

- `toggleHelper`

`persistedData` should be checked for `null` and default to a sane initial config. `notifcount:Number` should be `notifcount:number`.

```
Type '{ addresses: Json; popuptoggle: Number; }' is not assignable to type 'Record<string, Json>'.
  Property 'popuptoggle' is incompatible with index signature.
    Type 'Number' is not assignable to type 'Json'.
      Type 'Number' is not assignable to type '{ [prop: string]: Json; }'.
        Index signature for type 'string' is missing in type 'Number'.ts(2322)
```

snap/src/utils/toggleHelper.ts:L7-L16

```
let popuptoggle = notifcount;

const data = {
    addresses: persistedData.addresses,
    popuptoggle: popuptoggle,
};
await snap.request({
    method: 'snap_manageState',
    params: { operation: 'update', newState:data },
});
```

- `popupHelper`

`let msg = []` should be `let msg = [] as String[];`

```
Variable 'msg' implicitly has an 'any[]' type.ts(7005)
```

- `addresses` can be `null`

snap/src/utils/fetchnotifs.ts:L34-L37

```
export const fetchAllAddrNotifs = async () => {
    const addresses = await fetchAddress();
    let notifs:String[] = [];
    for(let i = 0; i < addresses.length; i++){
```

- `persistedData` can be `null`

snap/src/index.ts:L63-L68

```
let persistedData = await snap.request({
  method: "snap_manageState",
  params: { operation: "get" },
});

let popuptoggle = Number(persistedData.popuptoggle) + msgs.length;
```

Recommendation

Fix the typescript configuration (see issue 5.13). Fix all reported ts-lint errors. Avoid using `any` types and use safe types instead.

5.7 Avoid Hardcoding the Local Snap ID Minor

Description

The local snap-id is hardcoded in various places. Local snap IDs should not be used in production. Hence, we recommend defining and importing the snap id from a single source file within the project, setting it to `local:http://localhost:8080` and `npm:push-v1` depending on whether the build is set to be production or development (e.g., using an environment variable).

push-snap-site/components/buttons/ConfirmButton.tsx:L6-L8

```
export default function ConfirmButton() {
  const { address, isConnecting, isDisconnected } = useAccount();
  const defaultSnapOrigin = `local:http://localhost:8080`;
```

push-snap-site/components/buttons/ReconnectButton.tsx:L4-L6

```
export default function ReconnectButton() {

  const defaultSnapOrigin = `local:http://localhost:8080`;
```

push-snap-site/components/buttons/SendMessageButton.tsx:L1-L3

```
export default function ReconnectButton() {

  const defaultSnapOrigin = `local:http://localhost:8080`;
```

5.8 package.json - Invalid License Minor ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by changing the license to GPLv2.

Description

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

snap/package.json:L9

```
"license": "(MIT-0 OR Apache-2.0)",
```

Recommendation

Update the license field.

5.9 `fetchAddress` - Inaccurate Function Name Minor

Description

Function `fetchAddress` returns an array of addresses and should, therefore, be named `fetchAddresses`

snap/src/utils/fetchAddress.ts:L66-L73

```
export const fetchAddress = async () => {
    const persistedData = await snap.request({
        method: 'snap_manageState',
        params: { operation: 'get' },
    });
    const addresses = persistedData!.addresses;
    return addresses;
};
```

5.10 `currentepoch` - Unnecesary Conversion From/to String Minor ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by not converting `currentepoch` to string.

Description

It is unclear why `currentepoch` is declared as `String` while calculations require it to be numerical.

Examples

snap/src/utils/fetchnotifs.ts:L15-L31

```
export const filterNotifications=async(address:string)=>{
  let fetchedNotifications = await getNotifications(address);
  fetchedNotifications = fetchedNotifications?.feeds;
  let notiffeeds:String[] = [];
  const currentepoch:string = Math.floor(Date.now() / 1000).toString();
  if(fetchedNotifications.length > 0){
    for(let i = 0; i < fetchedNotifications.length; i++){
      let feedepoch = fetchedNotifications[i].payload.data.epoch;
      feedepoch = Number(feedepoch).toFixed(0);
      if(feedepoch > parseInt(currentepoch)-60) {
        let msg = fetchedNotifications[i].payload.data.app+' : '+fetchedNotifications[i].payload.data.amsg;
        notiffeeds.push(msg);
      }
    }
  }
  notiffeeds = notiffeeds.reverse();
  return notiffeeds;
```

Recommendation

`currentepoch` should be numerical.

5.11 Dead Code `popup` Minor ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by removing the used code.

Description

`const popup` is retrieved from the snap state but never used within the context of `confirmAddress()`. This might be an indicator of an incomplete implementation of the `togglePopup` setting or dead code.

snap/src/utils/fetchAddress.ts:L40-L47

```
export const confirmAddress = async () => {
    const persistedData = await snap.request({
        method: 'snap_manageState',
        params: { operation: 'get' },
    });
    const data = persistedData.addresses;
    const popup = persistedData.popuptoggle;
    let msg='';
```

Recommendation

Double check if this setting is meant to be read (unlikely) or else clean up and remove unused code.

5.12 Unused Import `ethers`, `@metamask/snaps-ui` Minor ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by using `ethers` to check if the provided address is well-formed and removing the `@metamask/snaps-ui` import from `popupHelper`.

Description

- `ethers`

`ethers` is listed as a dependency and imported by `fetchAddress.ts` but is never used.

snap/src/utils/fetchAddress.ts:L3

```
const {ethers} = require('ethers');
```

- `@metamask/snaps-ui`

`@metamask/snaps-ui` is imported in `popupHelper` but the imported components are never used.

snap/src/utils/popupHelper.ts:L1

```
import { heading, panel, text } from "@metamask/snaps-ui";
```

Recommendation

Remove the unused import/dependency.

5.13 Non-Existent Base Config (Eslint, Tsconfig) Minor

Description

`.eslintrc.js` points to a base configuration outside of this repository.

snap/.eslintrc.js:L2

```
extends: ['../../.eslintrc.js'],
```

- `.eslintrc.js`

```
⇒  npm run lint:eslint

> push-v1@0.1.0 lint:eslint
> eslint . --cache --ext js,ts


Oops! Something went wrong! :(

ESLint: 8.40.0

ESLint couldn't find the config "../../.eslintrc.js" to extend from. Please check that the name of the config is correct.

The config "../../.eslintrc.js" was referenced from the config file in "/Users/tintin/workspace/js/push-protocol-snaps/snap/.eslintrc.js".
```

- `tsconfig.json`

snap/tsconfig.json:L2

```
"extends": "../../tsconfig.json",
```

Recommendation

Provide the eslint base configuration with the repository to allow for reproducible lint runs. Run the linter as part of github commit checks.

5.14 Performance - `await` in for Loop ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by using `Promise.all()` instead.

Description

Performing an `await` as part of each operation is an indication that the program is not taking full advantage of the parallelization benefits of `async/await`:

snap/src/utils/fetchnotifs.ts:L38

```
let temp = await filterNotifications(addresses[i]);
```

Recommendation

Using `Promise.all()` fully utilizes parallelism and improves performance

5.15 API Design - Consider Using Consistent RPC Method Names ✓ Fixed

Resolution

fixed in ethereum-push-notification-service/push-protocol-snaps@1a6a32ef760088ca59f73e555f41b5b5d871f761 by changing the rpc method names as per recommendation.

Description

Consider using descriptive RPC method names with a distinct prefix, e.g. `pushproto_initialize`, `pushproto_addaddress`, `pushprotoc_togglepopup`:

snap/src/index.ts:L17

```
case "hello": {
```

snap/src/index.ts:L22

```
case "init": {
```

snap/src/index.ts:L36

```
case "togglepopup": {
```

Note that `init` can be called multiple times and is not initializing anything.

Appendix 1 - Files in Scope

#	Total	Code	Comment	ToDo	Name	Sha1
1	129	118			src/index.ts	b2488a3bf3f5fb3c4d9985b2b600cf4126c8b05e
2	73	66			src/utils/fetchAddress.ts	c7d084edac54ab91449bbcf189d9aacec75f45fe
3	42	39	1		src/utils/fetchnotifs.ts	0c65aed01976c4c87c09525bd9f206dbd28967e5
4	12	11			src/utils/popupHelper.ts	007b0daec3ec899ee950db8ce593aad38551c90c
5	17	15			src/utils/toggleHelper.ts	64cd0f3848c55d39e03e48c7eaef399f8a83795c
=====	=====	=====	=====	=====		
Σ	273	249	1	0		

Appendix 2 - Disclosure