Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Audit-Report Kyraview MetaMask Stellar Snap 02.2024

Cure53, Dr.-Ing. M. Heiderich, M. Pedhapati, B. Casaje, L. Herrera

Index

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Introduction

This document summarizes the findings of a source code audit and feature review conducted by Cure53 on the Kyraview MetaMask Stellar Snap and its codebase. The work, commissioned by Kyraview Ltd. in October 2023, was carried out in February 2024 (CW06). A total of eight days were dedicated to achieving the expected coverage for this project.

The assessment comprised two distinct work packages (WPs), as follows:

- **WP1**: Source code audits of Kyraview MetaMask Stellar Snap & codebase
- **WP2**: Snap- & chain-specific feature reviews of Kyraview MetaMask Stellar Snap

Cure53 received complete access to source code, test documentation, and any additional resources necessary for the evaluations. Utilizing a white-box methodology, a team of four senior testers managed the project's preparation, execution, and finalization. All groundwork was completed in late January and early February 2024 (CW05) to ensure a smooth examination process.

Communication throughout the engagement utilized a dedicated Slack channel shared by both the Kyraview and Cure53 teams, including all relevant personnel. Communication was efficient, questions were minimal, and the prepared scope proved clear and comprehensive, resulting in no significant roadblocks during the assessment. Cure53 provided frequent status updates and findings, including live-reporting for the single *Critical* severity vulnerability via the shared Slack channel.

Cure53 achieved comprehensive coverage over the WP1 and WP2 scope, identifying a total of six findings. Five of these were classified as security vulnerabilities, while the remaining ticket was considered a general weakness with lower exploitation potential. While the moderate number of findings might be viewed favorably, the presence of a *Critical* vulnerability (see ticket KYR-01-001) remains a significant concern. However, it is commendable that the Kyraview team has already promptly addressed this private key leakage circumstance during the assessment itself.

Overall, while areas for improvement exist to meet optimal security standards, Cure53 is confident that addressing all identified issues documented in this report will effectively achieve this goal in the near future. The following sections delve deeper into the scope, test setup, and available testing materials. Subsequently, the report lists all findings in chronological order of detection, starting with vulnerabilities and followed by general weaknesses. Each finding includes a technical description, a proof-of-concept (PoC) where applicable, and recommended mitigation or fix strategies.

Finally, the report concludes with Cure53's general impressions, offering insights into the perceived security posture of the reviewed Kyraview MetaMask Stellar Snap and codebase.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Pentests & code audits against Kyraview Stellar Snap codebase & build**
  - **WP1**: Source code audits against Kyraview MetaMask Stellar Snap & codebase
    - **Primary focus:**
      - General tests & attacks against browser add-ons, extension snap-ins, independently of specific use case as a crypto wallet snap.
    - **Sources:**
      - **URL:**
        - https://github.com/paulfears/StellarSnap
      - **Commit:**
        - abf73d844a85f42a706d73792ac1aeb28fa7ed90
      - **Version:**
        - 0.0.21
    - **Documentation:**
      - https://github.com/paulfears/StellarSnap/blob/main/README.md
  - **WP2**: Snap- & chain-specific feature reviews against Kyraview MetaMask Stellar Snap
    - **Primary focus:**
      - Specific features including (but not limited to):
      - Integration of Stellar blockchain network.
      - Non-custodial wallet features & security.
      - Key generation features & key handling / storage security aspects.
    - **Sources:**
      - *See WP1*.
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., *KYR-01-001*) to facilitate any future follow-up correspondence.

KYR-01-001 WP2: Malicious dApp can leak wallet private keys *(**Critical**)*

Fix note: *This issue has been mitigated by the Kyraview team and fix-verified by Cure53.*

Cure53 discovered that the *setCurrentAccount* RPC method leaks the entire snap's state to dApps. Consequently, a malicious dApp can access all user private keys and seeds.

This circumstance is enabled because the *currentState* value set via the *StateManager.getState()* function is returned when the aforementioned method is called, and is henceforth leaked to any website wherein the Stellar Snap is installed.

PoC:

```
ethereum.request({
    method: "wallet_invokeSnap",
    params: {
        snapId: "npm:stellar-snap", request: { method: "getCurrentAccount",
params: {} }
    }
})
.then((result)=>{
    ethereum.request({
        method: "wallet_invokeSnap",
        params: {
            snapId: "npm:stellar-snap",
            request: {
                method: "setCurrentAccount",
                params: { "address": result }
            },
        }
    })
    .then((result)=>{
        console.log(result);
    })
});
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Affected file:
/src/Wallet.ts

Affected code:

```
static async setCurrentWallet(address:string, currentState?:State,
setState?:Boolean){
    if(currentState === undefined){
        currentState = await StateManager.getState();
    }
    if(setState === undefined){
        setState = true;
    }
    if(currentState.accounts[address]){
        currentState.currentAccount = address;
        if(setState){
            await StateManager.setState(currentState);
        }
    }
    else{
        Utils.throwError("404", "account not found");
    }
    return currentState;
}
```

Steps to reproduce:

1. Ensure that the MetaMask extension is installed, then access *https://stellar-demo.netlify.app/*.
2. Click the *Connect* button and install the Stellar Snap.
3. Copy the JavaScript code from the PoC section and execute it on the demo website using DevTools.

To mitigate this issue, Cure53 recommends only returning the address of the newly-set wallet, rather than returning the entire snap state. Moreover, the implementation should explicitly request user permission to amend the current wallet, given that the aforementioned method could be called while the user is interacting with another dApp.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KYR-01-002 WP2: Markdown & control characters allowed in dialogs *(Medium)*

Cure53 observed that many of the dialogs shown in the snap display user input using the *text* method provided by the snap's UI. However, this process is problematic since the *text* method permits rendering control characters and Markdown. Notably, both newlines and bold text can be injected into these fields.

A malicious dApp is granted a number of potential input control methods that would be rendered via the *text* method, such as *signStr*. These could spoof different fields in the UI, potentially forcing a user into accidentally signing a message or confirming a transaction.

PoC:

```
await ethereum.request({
    method: 'wallet_invokeSnap',
    params: {snapId:`npm:stellar-snap`,
        request:{
            method: 'signStr',
            params:{
                challenge: "**The following is the message to be signed:**\
                    n\nsign data\n\n**Spoofed field**: x"
            }
        }
    }
})
```

To mitigate this issue, Cure53 advises enforcing that user input is never passed into the *text* method and alternatively utilizing the copyable field, which ignores Markdown and tag blocks.

KYR-01-003 WP2: Accounts renamable without confirmation *(Medium)*

The *renameAccount* RPC method permits dApps to amend the name associated with an account. However, the testing team noted that this method does not prompt for user confirmation and, as such, can be performed silently in the background. Accordingly, a malicious dApp could rename a user's wallet to generate confusion, which can also be magnified by leveraging the UI element spoofing strategy outlined in ticket KYR-01-002 simultaneously.

When running the *switchAccount* RPC method, the address name to be switched to appears in the confirmation dialog. If a malicious dApp has renamed the user's wallet to contain UI spoofing elements, the user may accept the prompt without noticing which wallet they were switching to.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

PoC

```
await ethereum.request({
    method: 'wallet_invokeSnap',
    params: {snapId:`npm:stellar-snap`,
        request:{
            method: 'renameAccount',
            params:{
                address: "address",
                name: "**Name:** Test account\n\n**Status:** not funded"
            }
        }
    }
})
```

To mitigate this issue, Cure53 suggests integrating a confirmation prompt for the *renameAccount* RPC method to assure that account renaming cannot occur covertly in the background.

KYR-01-004 WP2: dApp origin not displayed in Snap UI *(Medium)*

Testing verified that an origin indicating which dApp is requesting permission to perform a given action is not displayed in most of the Snap's dialogs. This raises concern since a malicious dApp could impersonate other trusted dApps. In addition, RPC calls can be initiated from within third-party iframes embedded inside the victim's page, including *signAndSubmitTransaction* requests used for transactions.

Furthermore, this pitfall is enhanced since the majority of websites serve ads from third-party iframes, even though they do not control their content. If adversaries were able to serve an advertisement on a domain leveraging the Stellar Snap, they would be able to trigger RPC calls. From there, given that no origin is displayed in the Snap's UI, successful user tricking would be plausible. Thus, one could assume that the request has originated from a trusted website.

To mitigate this issue, Cure53 recommends creating a default template that clearly displays the origin that performed the RPC call, and including it in all dialogs utilized by the Snap component. One must acknowledge that the origin is available in the *onRpcRequest* handler.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

KYR-01-006 WP2: *clearState* default wallet alteration without confirmation *(Low)*

The observation was made that explicit user confirmation is required for the current wallet to be modified. This is necessary because a malicious dApp could amend the current wallet while the user is interacting with a different dApp, a process that could be leveraged to trick users into performing undesirable actions via alternate wallets.

Upon further investigation, the test team discovered that the *clearState* RPC method clears the entire state without any user confirmation. This later automatically sets the current wallet as the default via the *createNewAccount* function, which could potentially confuse the user.

Affected file:
/src/Wallet.ts

Affected code:

```
static async createNewAccount(name, currentState?:State,
setState?:boolean):Promise<Wallet>{
    [...]
    let tempAccount = await Wallet.getTempAccountFromSalt(salt);
    tempAccount.name = name;
    currentState.accounts[tempAccount.address] = tempAccount;
    if(currentState.currentAccount === null || numAccounts === 0){
        currentState.currentAccount = tempAccount.address;
    }
    if(setState){
        await StateManager.setState(currentState);
    }
}
```

To mitigate this issue, Cure53 recommends prompting the user for confirmation before clearing the snap's state via dialog usage, as well as clearly displaying the origin of the dApp that requested the method in the snap's dialog.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, while a vulnerability is present, an exploit may not always be possible.

KYR-01-005 WP1: Lack of parameter validation for RPC requests *(Low)*

While analyzing the *OnRpcRequestHandler* function responsible for handling all RPC requests sent by dApps, the audit team determined that parameter validation is not performed on the user-controlled input received. This can induce certain security ramifications, given that malicious input can be leveraged to bypass certain checks and cause the Stellar Snap to misbehave. One example of this pertains to the *renameWallet* function, which takes a user-controlled address value and is subsequently used to directly access an object via the bracket notation property accessor.

Due to the fact that the address is not validated, it can assume values such as *__proto__* and *constructor*, which are native JavaScript properties that all objects inherit. These would pass the conditional check, even though the wallet object would be invalid. Fortunately, the aforementioned behavior does not induce security issues at present, though the Kyraview team should address it to definitively rule out the possibility of such shortcomings arising in the future.

Affected file:
/src/Wallet.ts

Affected code:

```
static async renameWallet(address, name,
currentState?:State):Promise<boolean>{
    if(!currentState){
        currentState = await StateManager.getState();
    }
    let wallet = currentState.accounts[address];
    if(!wallet){
        Utils.throwError(900, "account not found");
        return false;
    }
    wallet.name = name;
    currentState.accounts[address] = wallet;
    await StateManager.setState(currentState);
    return true;
}
```

To mitigate this issue, Cure53 recommends performing strict validation against all user-controlled params in the *OnRpcRequestHandler* prior to calling their respective functions.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

This report's *Conclusions* section aims to offer an accurate appraisal of the scope's security performance, based on the advanced technical analysis and resulting coverage performed by Cure53 during this Q1 2024 exercise. In sum, the relatively low yield of findings garnered a favorable impression, though the various hardening opportunities should be addressed as soon as possible for airtight defense against possible breaches.

Cure53's analysis of the Snap manifest revealed adherence to security best practices established by MetaMask Snaps' guidelines. By requesting only the essential permissions, the Stellar Snap effectively minimizes its attack surface, reducing potential security risks.

Comprehensive reviews and exploitation attempts targeted exposed RPC methods within the codebase to maximize coverage and uncover vulnerabilities. This rigorous process identified a *Critical* issue wherein malicious dApps could exploit the Snap to leak private keys and seeds, as emphasized in ticket KYR-01-001.

During WP1, the security audit focused on identifying general vulnerabilities beyond the Stellar Snap's crypto wallet functionality. The smaller inherent attack surface of MetaMask Snaps compared to traditional browser add-ons resulted in few findings in this category. Consequently, the majority of issues identified within this report pertain specifically to features associated with the Snap's crypto wallet capabilities.

Moving on, Cure53's examinations revealed a certain deficiency within the *clearState* RPC method, as discussed in ticket KYR-01-006. This method lacks user confirmation, permitting malicious dApps to reset the wallet value to its default state without their knowledge. This exploitation poses significant risks, potentially leading to the compromise of user funds and undesirable outcomes while interacting with other dApps.

Additionally, the observation was made that the Snap fails to perform strict parameter validation on its content, thereby enabling unexpected parameter values to be passed into the Snap's functions and evoking unintended consequences, as documented in ticket KYR-01-005.

With respect to storage, the utilization of *snap_manageState* to store sensitive user data was deemed an astute solution from a security perspective. This is because the stored contents are automatically encrypted using a Snap-specific key and automatically decrypted when retrieved. It is important to highlight that all dApps share the same state by design and are not compartmentalized by origin. This should be heeded when interacting with the snap state, since leakage will lead to a compromise of the user's private keys, as indicated in ticket KYR-01-001.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

While the Snap utilizes the *endowment:network-access* permission, granting attackers the ability to execute and receive responses to requests within the Snap, no security vulnerabilities associated with this functionality were identified due to the secure sandboxing environment. Additionally, reviews of external request functions confirmed the absence of path traversal vulnerabilities, mitigating the potential risk of unauthorized or partial requests executed on behalf of the Snap.

During subsequent analysis of the Snap's UI components, Cure53 identified a vulnerability regarding the omission of dApp origin information in most Snap dialogs. This design flaw enables malicious dApps to exploit user trust by masquerading as reputable entities, potentially manipulating users into performing unauthorized actions (see ticket KYR-01-004).

Further analysis of the Snap's UI revealed a security weakness pertaining to user input. The audit team’s findings indicate that many Snap dialog and confirmation prompts permit the inclusion of Markdown and control characters within user-provided information. This subpar activity may allow malicious dApps to manipulate UI elements within dialog boxes, potentially leading to user confusion and inadvertently triggering unintentional actions. For supporting guidance, please refer to ticket KYR-01-002.

Elsewhere, the assessors located another shortcoming in the *renameAccount* RPC method whereby the user is not prompted for confirmation before renaming their account (see ticket KYR-01-003). The fact that this occurs covertly in the background can be problematic, since the victim user may not realize that their account name has been substituted. Moreover, their account may be switched with another entirely, without their knowledge. In addition, the previously described UI spoofing drawback could be utilized to exacerbate user confusion in this context.

While the security audit identified a limited number of vulnerabilities, the presence of a *Critical* issue amongst them necessitates improvement. This mixed outcome suggests that the Stellar Snap is bolstered with a decent security stance at present, but further measures are required to achieve a first-rate security standard. Addressing and mitigating all findings documented in this report should be considered essential steps towards solidifying the Snap's overall security posture.

Cure53 would like to thank Paul Fears of the Kyraview Ltd. team for his excellent project coordination, support, and assistance, both before and during this assignment.