Fine penetration tests for fine websites

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

Pentest-Report WalletChat MetaMask Snap 07.2023

Cure53, Dr.-Ing. M. Heiderich, M. Pedhapati

Index

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

Introduction

"Turn any dapp social. Chat wallet-to-wallet. Market-leading web3 messaging. Connect with others directly, using your wallet. Integrate inside your dapp via our out-of-the-box widget or API."

From https://www.walletchat.fun/

This report describes the results of a security assessment of the WalletChat MetaMask Snap, with focus on its codebase and integration. The project, which included a penetration test and a dedicated source code audit, was carried out by Cure53 in July 2023.

Registered as *WAC-01*, the examination was requested by WalletChat Labs, Inc in June 2023 and then scheduled to start the following month, i.e., in July 2023. Since this project marked the first cooperation between Cure53 and WalletChat, it was important for both sides to have sufficient time to prepare.

In terms of the exact timeline and specific resources allocated to *WAC-01*, Cure53 completed the research in early July 2023, namely in CW27. In order to achieve the expected coverage for this task, a total of two days were invested. In addition, it should be noted that a team of two senior testers was formed and assigned to prepare, execute, and deliver this project.

For optimal structuring and tracking of tasks, the examination was split into two separate work packages (WPs):

- **WP1**: Source code audits against WalletChat MetaMask Snap & codebase
- **WP2**: Code audits & feature reviews against WalletChat MetaMask Snap & integration

As can be deduced from the formulations of the WP titles, white-box methodology was utilized. Cure53 was provided with documentation, a list of focus items, as well as all further means of access required to complete the tests. Additionally, all sources corresponding to the test-targets were shared to make sure the project can be executed in line with the agreed-upon framework.

Overall, the project progressed effectively. To facilitate a smooth transition into the testing phase, all preparations were completed in CW26. Throughout the engagement, communications were conducted via a private, dedicated and shared Slack channel. Stakeholders - including the Cure53 testers and the internal staff from WalletChat Labs - could participate in discussions in this space.

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

The quality of the interactions throughout the test was excellent, with no outstanding queries. These steady exchanges contributed positively to the overall outcomes of this project. The scope was well prepared and clear, which played a major role in avoiding significant roadblocks during the test. Cure53 also offered frequent status updates about the test and the emerging findings.

The Cure53 team succeeded in achieving very good coverage of the WP1-WP2 scope items. Of the seven security-related discoveries, three were classified as security vulnerabilities and four were categorized as general weaknesses with lower exploitation potential. It should be noted that the total number of problems is moderate, which indicates that the tested MetaMask components are already effective in defending against various attacks and major risks.

Nevertheless, it is important to highlight one of the findings which was ranked as a *High*-level risk. This problem demonstrates a scenario, in which an arbitrary DApp can retrieve the *access* token (see WAC-01-001). It is recommended to swiftly resolve this issue in order to ensure that good levels of security can be maintained.

The following sections first describe the scope and key test parameters, as well as how the WPs were structured and organized. Next, all findings are discussed in grouped vulnerability and miscellaneous categories. Flaws assigned to each group are then discussed chronologically. In addition to technical descriptions, PoC and mitigation advice will be provided where applicable.

The report closes with drawing broader conclusions relevant to this July 2023 project. Based on the test team's observations and collected evidence, Cure53 elaborates on the general impressions and reiterates the verdict. The final section also includes tailored hardening recommendations for the WalletChat MetaMask Snap complex, with the focus on its codebase and integrations.

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

Scope

- **Source code audits & security reviews of WalletChat MetaMask Snap & Codebase**
  - **WP1**: Source code audits against WalletChat MetaMask Snap & codebase
    - **Focus areas:**
      - General tests & attacks against browser add-ons, extension Snap-ins, independent of the specific crypto-wallet snap use-case
    - **Source:**
      - https://github.com/Wallet-Chat/walletchat-metamask-snap
    - **Documentation:**
      - https://docs.walletchat.fun/metamask-integration
      - https://snaps.walletchat.fun
    - **Audited commit**
      - fec67c2a588fb9db132501b70165e8ef75d100ec
  - **WP2**: Code audits & feature reviews of WalletChat MetaMask Snap & integration
    - **Focus areas:**
      - Specific features including - but not limited to - chat features, message management and delegation, user and chat partner discovery, possible information leaks via chat and message handling, authentication & login.
    - **Sources and documentation:**
      - *See WP1*
    - **Audited commit:**
      - fec67c2a588fb9db132501b70165e8ef75d100ec
  - **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, each ticket is given a unique identifier (e.g., *WAC-01-001*) to facilitate any future follow-up correspondence.

WAC-01-001 WP2: Arbitrary DApp can retrieve *access* token *(High)*

***Fix note**: The bug was fixed during the testing phase and the Cure53 team validated the proposed fix.*

***Fix commit hash:** c63cfc27ecd8fdb1ca38d8391aeb298c0a3f6ab4*

While auditing the SIWE flow, it was noticed that arbitrary DApps can retrieve *access* tokens for WalletChat. This is due to the endpoint at */users/{public_key}/nonce,* which allows retrieval of *nonce* for a specific public key of a wallet without any authentication. Moreover, the sign-in endpoint allows any valid signed message. This means that an arbitrary DApp can request a signature for an arbitrary message using *personal_sign* RPC. From there, it can use the signature and the *nonce* to retrieve the *access* token from the backend. Using the token, an adversary can retrieve messages from the *walletchat* user.

Below is a request to the API endpoint with the *nonce* retrieved without any authentication for a specific public key and a signature for arbitrary message *"asdf"*.

Request

```
POST /signin HTTP/2
Host: api.v2.walletchat.fun
Content-Length: 270
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

{"name":"1","address":"0x899813CF950E6fa211fe600dEEEcfb1849427B47","nonce":"8917
0973333945986509035566844483179635 1","msg":"asdf","sig":"0x08e4128fa14de93fd709c
2b551ac1c6d47a93387fc70179c14d0fed756270dbd3c729762b49a0559f61b978a4b2353f24b028
8a985fb600f04682f92731d1dca1b"}
```

It is recommended to fix the vulnerability by tying the *nonce* to a specific session and, more importantly, validate the signed message to be in a SIWE format. The latter should contain WalletChat domain, *nonce* and other information instead of arbitrary messages in the backend. This means a user can judge if the signed message is coming from the valid WalletChat domain before signing, which is because the MetaMask will show a deceptive domain's warning in the pop-up.

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

WAC-01-002 WP1: Restrict Snap RPC access to trusted origins *(Medium)*

***Fix note**: The bug was fixed during the testing phase and the Cure53 team validated the fix.*

***Fix commit hash**: 1c7385d3cd2728603ca3a2daab05934809f69c79*

Testing confirmed that the *onRpcRequest* handler failed to validate the origin of the request. This effectively signifies connections to the WalletChat Snap instance from arbitrary websites which could then send RPC requests. A connected website can set the Snap state with an adversary-controlled account's *apiKey* using *set_snap_state* method.

As a consequence, an adversary can log into their account via the victim's MetaMask Snap, hence masquerading as a legitimate victim's account. More importantly, the MetaMask Snap's popup does not include any information regarding the currently logged-in user, as shown in the below figure. This also facilitates obfuscation and may clearly make it easier to successfully perform phishing-related attacks.

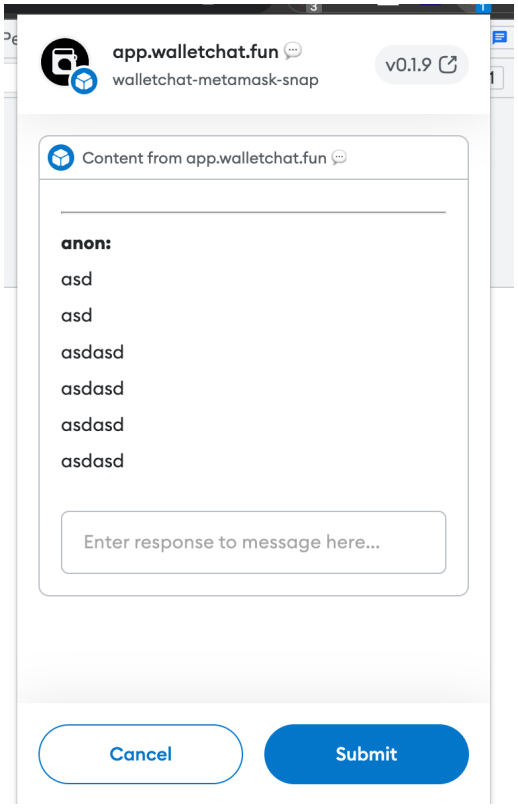

Fig.: MetaMask WalletChat Snap notification popup masqueraded as victim's account.

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

Affected file:
walletchat-metamask-snap/packages/snap/src/index.ts

Affected code:
```
export const onRpcRequest: OnRpcRequestHandler = async ({ request }) => {
  switch (request.method) {
[...]
 case 'set_snap_state':
      if (
        (request.params &&
        'apiKey' in request.params &&
        typeof request.params.apiKey === 'string') &&
        request.params &&
        'address' in request.params &&
        typeof request.params.address === 'string'
      ) {
        await setSnapState(request.params.apiKey, request.params.address);
        return true;
      }
[...]
```

It is recommended to fix the vulnerability by validating the origin. Only specific domains related to the WalletChat application should be allowed in this context.

WAC-01-006 WP2: Unsafe wild card *targetOrigin* usage in *postMessage* *(Low)*

The WalletChat Snap Site has an insecure *postMessage* wild card, which lets a web application embed the site and leak information using the *postMessage* handler. However, it was noticed that the *postMessage* did not leak any sensitive data through this component, which explains a reduced impact being ascribed to this problem.

PoC:
https://ctf.s1r1us.ninja/xss.php?code=<embed src="https://snaps.walletchat.fun/" style="width:100%; height: calc(100vh - 180px); border: 1px solid lightgrey;"> <script>window.onmessage=function(e){if(e.origin=='https://snaps.walletchat.fun') {console.log(e.data)}} </script>

Affected file:
src/context/UnreadCountProvider.js

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

Affected code:

```
[...]
  let msg = {
                    "data": total_cnt,
                    "target": "unread_cnt"
                }
                window.parent.postMessage(msg, "*"); //targertOrigin should be
a .env variable
[...]
  window.parent.postMessage(
            {
               data: 1,
               target: 'unread_cnt',
            },
            '*' // targertOrigin should be a .env variable
        )
```

To fix this vulnerability, a specific origin in *targetOrigin* should be used instead of using the wild card. Furthermore, it is also recommended to use the *X-Frame-Options* header to disallow embedding of the Snap site.

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, whilst a vulnerability is present, an exploit may not always be possible.

WAC-01-003 Web: General HTTP security headers missing on Snap Site *(Low)*

It was found that the WalletChat Snap Site platform is missing certain HTTP security headers in HTTP responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other problems. For instance, a malicious site can frame the *snaps.walletchat.fun* and perform clickjacking attacks. The following list enumerates the headers that need to be reviewed to prevent flaws linked to headers.

- ***X-Frame-Options***: This header specifies whether the web page is allowed to be framed. Although this header is known to prevent Clickjacking attacks, there are many other attacks which can be achieved when a web page is frameable[1]. It is recommended to set the value to either **SAMEORIGIN** or **DENY**.
- Note that the CSP framework offers similar protection to X-Frame-Options in ways that overcome some of the shortcomings of the aforementioned header. To optimally protect users of older browsers and modern browsers at the same time, it is recommended to consider deploying the *Content-Security-Policy: frame-ancestors 'none';* header as well.
- ***X-Content-Type-Options***: This header determines whether the browser should perform MIME Sniffing on the resource. The most common attack abusing the lack of this header is tricking the browser to render a resource as a HTML document, effectively leading to Cross-Site Scripting (XSS).

Overall, missing security headers is a bad practice that should be avoided. It is recommended to add the aforementioned headers to every server response, including error responses like *4xx* items.

More broadly, it is recommended to reiterate the importance of having all HTTP headers set at a specific, shared and central place rather than setting them randomly. This should either be handled by a load balancing server or a similar infrastructure. If the latter is not possible, mitigation can be achieved by using the web server configuration and a matching module.

[1] https://cure53.de/xfo-clickjacking.pdf

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

WAC-01-004 Web: Lack of *Content-Security-Policy* header on Snap Site *(Low)*

Testing confirmed that the WalletChat Snap Site application did not currently utilize the *Content-Security-Policy*[2] (CSP) HTTP response header. This signifies that associated benefits of this header are not leveraged by the complex, even though it provides additional defense-in-depth by allowing definition of policies for certain HTML tags, including *script* elements.

In addition, the CSP header enables specifying the origin from which a resource can be loaded. The primary purpose of this header is to prevent or significantly impede malicious HTML injections. An XSS in Snap Site would mean direct interaction with MetaMask WalletChat Snap RPC methods, therefore, it is important to have CSP deployed to further strengthen the Snaps.

To address this issue, Cure53 recommends implementing the CSP header for the WalletChat Snap Site application. It is important to note that an effective CSP ruleset can only be achieved through the strictest configuration possible, especially in terms of the handling of JavaScript execution with the *script-src* directive. Depending on the complexity of the web applications currently in use, this may require significant engineering effort, as integrated CSP rules have the potential to disrupt benign aspects of an application, and therefore require careful review prior to deployment.

WAC-01-005 WP2: Unsanitized URLs passed to *anchor* tag *href* attribute *(Low)*

While auditing the frontend JavaScript for DOM-based XSS, it was noticed that the *external_url* and the *discord_url* in the NFT record were not sanitized before being assigned to the *anchor* tag *href* attribute. This leads to XSS if these links contain dangerous JavaScript protocol URLs.

The following code snippets show the affected code where the *collectionLInk* and *sourceLink* are added to the *anchor* tag without sanitization.

Affected file #1:
src/scenes/NFT/scenes/NFTByContract/NFTByContract.tsx

Affected code #1:

```
[...]
  {nftData?.external_url && (
                          <Tooltip label="Visit website">
                             <Link
                                href={nftData.external_url}
```

[2] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

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

```
                              target="_blank"
                              d="inline-block"
                              verticalAlign="middle"
                              mr={1}
                            >
[...]
     {nftData?.discord_url && (
                          <Tooltip label="Discord">
                             <Link
                                href={nftData.discord_url}
                                target="_blank"
                                d="inline-block"
                                verticalAlign="middle"
                                mr={1}
[...]
```

Affected file #2:
src/scenes/NFT/scenes/POAPById/POAPById.tsx

Affected code #2:

```
[...]
<Button
                             size="xs"
                             href={poapEvent?.event_url}
                             as={CLink}
                          >
[...]
```

To fix this vulnerability, the URL should be sanitized and only HTTP(S) protocol links should be permitted.

WAC-01-007 WP1: Client-side path traversal in Snap fetch requests *(Info)*

The WalletChat MetaMask Snap utilizes user-controlled *address* variables to craft HTTP API requests. During the assessment two instances were detected with these variables and could be leveraged to create an HTTP path without applying any validation.

A would-be attacker could, on this basis, craft a URL that modifies the intended API endpoint and targets an arbitrary endpoint instead. It is important to note that the *address* variable can only be modified by a connected site via *set_snap_state* RPC. Notably, this behavior does not introduce a security issue in isolation, but could prove highly beneficial in the context of exploiting more significant attack vectors and web vulnerabilities.

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

Affected file:
packages/snap/src/index.ts

Affected code:

```
[...]
const getUnreadCountFromAPI = async (apiKey: string, address: string) => {
  let retVal = 0

  await fetch(
    `https://api.v2.walletchat.fun/v1/get_unread_cnt/${address}`,

[...]
const getLastUnreadMessage = async (apiKey: string, address: string) => {
  let chatData = ''

  await fetch(
    `https://api.v2.walletchat.fun/v1/get_last_unread/${address}`,
    {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${apiKey}`,
      },
    }
```

To mitigate this issue, one can recommend validating *address* variables before using them in subsequent HTTP paths. It can be considered that the addresses are alpha-numeric only which, generally speaking, means they should never contain any URL-relevant characters such as *"./#?&"*. Hence, simply checking the specified *address* to ensure that it is alpha-numeric would be sufficient.

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

Conclusions

Cure53 concludes that the codebase and integrations of the WalletChat MetaMask Snap project demonstrate an already well-secured foundation. However, the findings from this July 2023 examination also showcase that there is still room for improvement on the tested scope. Some more work and effort need to be invested by the WalletChat team in order to achieve an excellent level of security.

The testing started off by Cure53 auditing the Snap manifest and configuration for insecure configuration patterns, including overly lax permissions. The testing team concluded that the Snap element only requests required permissions and no misconfigurations could be spotted during this *WAC-01* investigation.

Despite the highly constrained attack surface, the Cure53 team evaluated every exposed RPC method accessible in an attempt to uncover any negative security implications. No major issues were identified in the RPC handlers, but a minor issue revealed that the *address* variables were not sanitized before being appended to the path of the request made from fetch API (see WAC-01-007).

Due to the nature of the WalletChat Snap application, the team noticed that the Snap should not allow RPC requests from all origins, as this makes it possible for attackers to masquerade as legitimate users in the notifications. More details on this can be found in WAC-01-002.

Additional checks for issues associated with the interaction between unauthorized DApps and Snap were conducted, but all attempts failed. The approaches of the testers were, in other words, successfully prevented by MetaMask's security checks.

Moving on to the Snap Site component, the team conducted thorough checks in the frontend, focusing on XSS vulnerabilities. This is because such flaws could directly influence Snap from the site. It was then assessed whether any client-side-related security issues associated with XSS, *postMessage*, and prototype-pollution could be located.

Toward this, the testing team noted that the majority of the frontend utilizes the ReactJS framework, which features a well-tested escaping mechanism. As such, it prevents many XSS-related issues by default. Since the ReactJS framework does not supervise the URLs assigned to the *href* property of the HTML *anchor* tags, the source code was searched for these issues. This led to the identification of WAC-01-005, as well as several miscellaneous issues documented in WAC-01-003, WAC-01-006 and WAC-01-004.

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

In addition, risk-laden instances of *window.open, window.location* and usage of similar items were inspected. Positively, no associated findings were identified in this area, which corroborates the strong impression gained by the add-on. Lastly, the team concentrated on looking for issues in the authentication of the Snap via Sign-In with Ethereum mechanism. It was hoped that misconfigurations or vulnerabilities would be spotted, but only the issues mentioned in WAC-01-001 were specified.

In conclusion, following the completion of this *WAC-01* security audit, Cure53 garnered a positive impression over the security premise established by the WalletChat MetaMask Snap installed in the MetaMask. However, the Snap Site appears to be slightly less secure and this viewpoint is corroborated by the discovery of the *High* impact issue, alongside some miscellaneous issues. Once these issues are fixed, the whole WalletChat MetaMask Snap can more consistently accomplish a better-protected security posture.

Cure53 would like to thank Kevin Larson and Michal Kubis from the WalletChat Labs, Inc team for their excellent project coordination, support and assistance, both before and during this assignment.