Fine penetration tests for fine websites

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

Pentest-Report Trust Wallet Browser Addon 04.2023

Cure53, Dr.-Ing. M. Heiderich, M. Pedhapati, L. Herrera, L. Hu

Index

- Introduction
- Scope
- Test Methodology
- Identified Vulnerabilities
  - BIN-04-001 FIXED WP1: SOP bypass on specific JSON formats (High)
  - BIN-04-002 FIXED WP1: Popup undismissed on client-side redirects (Low)
  - BIN-04-003 FIXED WP1: CSP's sandbox directive facilitates Null origin (Low)
  - BIN-04-004 FIXED WP1: Lack of optimal watchAsset validation incurs DoS (Low)
  - BIN-04-005 FIXED WP1: Lack of addEthereumChain type validation incurs DoS (Low)
- Conclusions

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

Introduction

"The Trust Wallet Browser Extension is your secure crypto wallet and gateway to thousands of Web3 dApps. Swap tokens, play games, earn rewards, and more."

From https://trustwallet.com/browser-extension

This report entitled BIN-04 documents the scope, test methodology, findings, and conclusions of a penetration test, source code audit, and feature review against the Trust Wallet Browser Extension. This security review was conducted by four skill matched senior members of the Cure53 team over the course of 10 days in CW14 April 2023, as requested by DApps Platform Inc. in January 2023. This audit time frame was defined in advance to fulfill the expected coverage expectations.

To provide some context regarding previous collaborations between the two organizations, the Trust Wallet Browser Extension has been analyzed by Cure53 on one previous occasion. Specifically, the extension represented the primary scope item for an assessment held in December 2022 and denoted under report BIN-03. For execution efficiency, all test-related actions were grouped into two separate work packages (WPs), as defined by the following headings:

- **WP1**: Source code audits against Trust Wallet browser extension
- **WP2**: Code audits and feature reviews against Trust Wallet browser extension

The client provided sources, assisting documentation, and any other means of access required by the test team to conduct the audit. As one can deduce from the WPs outlined above, the selected penetration testing methodology for this particular engagement was white-box. These materials were leveraged to complete any necessary preliminary preparations shortly before the assessment phase, namely in CW13 2023. This helped to ensure seamless coverage over the components in scope.

A dedicated, shared Slack channel was established for communication purposes between DApps Platform Inc. and Cure53. All active personnel from both parties relevant to this particular project were invited to join the channel and engage in cross-team discussions.

In general, communications were carried out only when absolutely necessary; the test team encountered little difficulties and as such rarely relayed any queries or requirements to the client. This partly owed to the transparent and effective scope preparation, which also ensured that no noteworthy delays or blockers would be experienced.

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

Another specific test-supporting facet, live reporting, was offered by Cure53 and subsequently achieved via the aforementioned Slack channel. During this process, the audit team relayed a number of updates concerning the findings and assessment progress in general.

With reference to the findings, Cure53 is pleased to report that a satisfactory degree of coverage across the WP1 and WP2 scope items was achieved for this pentest. In total, the test team confirmed the presence of five findings, all of which were categorized as security vulnerabilities and subsequently assigned to the *Identified Vulnerabilities* section outlined later in this report.

The total yield of findings was considered minimal, which in itself reflects considerably favorably on the Trust Wallet Browser Extension's security implementation. Furthermore, the vast majority of the weaknesses detected exhibit minor exploitation potential and were consequently allocated a *Low* severity marker. However, this should not detract from the *High* severity issue addressing a potential SOP bypass, as stipulated in ticket BIN-04-001. Cure53 strongly advises adhering to the guidance offered and resolving the flaw at the earliest possible convenience, which will guarantee that the extension offers steadfast robustness against any bypass scenario.

In summation, Cure53 is pleased to conclude that the Trust Wallet Browser Extension incorporates performant security paradigms to negate an array of potential security concerns. This viewpoint also correlates with the positive development observed in comparison with the preceding audit. Nevertheless, the *High* severity vulnerability remains a prime example of the need to invest ample and continuing resources into strengthening the extension. This would ensure that a top-level and trustworthy security offering can be maintained for the platform and its user base.

The report will now outline the scope and setup, as well as refer to the provision of test materials and sources, in the bullet points defined below. This section is followed by the *Test Methodology* chapter, which provides a definitive overview of the advanced techniques employed by Cure53 in its efforts to achieve maximum coverage and yield of findings. All in-scope characteristics are addressed, with commentary regarding the method by which they were examined and the reasoning behind the test team's endeavors. Essentially, this section serves to counterbalance the lack of a substantial volume of findings, despite the auditors' rigorous efforts.

Next, all findings are listed in chronological order of detection, starting with the *Identified Vulnerabilities* and culminating with the *Miscellaneous Issues* (albeit none representing the latter were encountered here).

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

Each finding offers a technical description, a Proof-of-Concept (PoC) if required, and the proposed mitigation guidance relevant to the specific context.

In conclusion, Cure53 verifies its viewpoint of the Trust Wallet Browser Extension's security posture by discussing the general impressions gained and security findings unearthed throughout the course of this test iteration.

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

Scope

- **Source code audits & security reviews against Trust Wallet Browser Extension**
  - **WP1**: Source code audits against Trust Wallet Browser Extension, JS & TS
    - **Sources:**
      - *extension-wallet-1.0.18.zip*
    - **Extension build:**
      - */release-1.0.18-0cc296a*
  - **WP2**: Code audits & feature reviews against Trust Wallet Browser Extension
    - **Primary focus area (not exclusive):**
      - Multi-Wallet support
      - Ledger & Posthog integration
    - **Sources:**
      - *extension-wallet-1.0.18.zip*
    - **Extension build:**
      - */release-1.0.18-0cc296a*
  - **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

Test Methodology

The following passages provide an overview of the testing methodology and subsequent coverage achieved during this engagement against the Trust Wallet Browser Extension and associated characteristics. All in-scope features were meticulously scrutinized via advanced penetration techniques by the Cure53 team, as stipulated below. In essence, the *Test Methodology* section serves to define Cure53's comprehensive assessment procedures, despite the relatively minimal findings encountered.

- In context, the Trust Wallet UI and frontend implementation are based on the ReactJS framework, which inherently provides effective built-in mitigation against XSS scenarios. Since XSS may still be plausible through usage of *dangerouslySet** functions, Cure53 audited the provided sources for any employment of these functions throughout the framework.
- The source code analysis for any risk-inducing calls was conducted in tandem with dynamic testing, which was achieved by providing malicious input into various input fields. Cure53's efforts in this respect verified a lack of unintended behaviors, thereby no security concerns were identified. To confirm this, the team leveraged tools such as DOM Invader[1] and postMessage-tracker[2].
- The *redirect* parameter was noted to be retrieved directly from the query string and passed to the navigate function in the Wallet popup page. Following review, this implementation was deemed sufficiently secure.
- Next, the extension's manifest file was probed, which validated that a limited set of files were exposed to the internet via the *web_accessible_resources* property. This was positively acknowledged by the test team, since a greater volume of file exposure would expand the connected attack surface. The developer team's strict CSP integration also garnered a favorable impression, helping to minimize any potential exploitation of XSS issues.
- The *content.js* and *inpage.js* files were extensively audited due to their functional significance of being injected into all web pages and utilized as a bridge for the communication between the page and extension. Positively, no security risks were detected.
- In relation to passphrase leakage, Cure53 noted that the extension popups are reloaded every time the wallet is locked, which prevents the password and the secret key from remaining in memory. This is evidently an improvement in comparison with previous assessment, since related issues were previously detected that have been subsequently nullified.
- The extension's attack surface in general was stringently inspected and considered compositionally small, primarily due to the extensive robust and

[1] https://portswigger.net/burp/documentation/desktop/tools/dom-invader
[2] https://github.com/fransr/postMessage-tracker

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

established communication-related functionality utilized for handling message exchanges between the background, content script, and web pages. The separation between INTERNAL and PUBLIC methods was deemed consistent and does not incur any detrimental security implications.

- In addition, the middlewares implemented by the background script were examined in-depth, which raised the presence of a high-severity issue leading to a partial SOP bypass, as stipulated in ticket BIN-04-001. Notably, this vulnerability represents a bypass of a previously resolved flaw.
- The test team placed specific emphasis on scanning the new Multi-Wallet feature, with a multitude of testing approaches initiated to ascertain its security resilience. Here, all attempts to confuse the UI into displaying a different wallet to that which was currently active were unfruitful. Similarly, race condition attacks were successfully repelled. Cure53 instigated attempts to directly specify wallets that were not connected to RPC calls that accept arbitrary addresses, whilst edge cases concerning switching wallets and multiple wallets connected to the same dApp were reviewed.
- Elsewhere, another key area of focus was the Posthog integration and all associated code. Here, Cure53 noted that the *autocapture* option is disabled; a sound design choice, considering its ability to prevent sensitive information being sent to the analytics server in error. Pertinently, should this option be enabled in a future implementation, the extension would need to use the ph-no-capture class for the relevant fields.
- Similarly, the code related to the Ledger integration was deep-dive inspected, for which the test team was unable to detect any associated weaknesses during the time frame of the audit.
- An analysis of the alternating security indicators and the method by which they are reflected in the Trust Wallet UI was performed. In this regard, a bypass of a previous related issue was identified and documented in ticket BIN-04-003. A correlatory UI flaw pertaining to the dismissal of the popup window during render-initiated navigation was also uncovered, as addressed in ticket BIN-04-002.
- Finally, the parameter validation and how this consequently affects the UI was researched. Cure53 verified that some basic sanity checks have been established, though further validation measures should be incorporated. Failure to optimally validate parameters may otherwise facilitate DoS issues, due to the fact that unexpected data types could evoke UI crashes. Ultimately, two low-severity DoS issues were detected; additional guidance on these is offered in tickets BIN-04-004 and BIN-04-005 respectively.

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., *BIN-04-001*) to facilitate any future follow-up correspondence.

BIN-04-001 FIXED WP1: SOP bypass on specific JSON formats *(High)*

***Note**: This issue was fixed by the Trust Wallet team and the fix was verified by Cure53 via inspecting a diff. The problem as described no longer exists.*

The observation was made that the *eth_sendRawTransaction* method facilitates the ability to send a transaction to a custom RPC URL without any user validation or confirmation. An attacker could leverage the *eth_sendRawTransaction* to send a request to a custom malicious RPC URL endpoint and thus successfully set up the fetch middleware, rendering the deployed *getChainId* verification successful. Further POST/GET requests to this URL may be redirected to a cross-origin page, thereby permitting an attacker to read the page's response and subsequently bypass the Same-Origin Policy (SOP).

This issue was previously reported and mitigated in the prior pentest (BIN-03-005). However, following supplementary investigations, Cure53 verified the possibility to bypass the current validation. This can be achieved by initially returning a valid response that reflects the *id* value sent in the body of the initial POST request, then redirecting the page to the arbitrary URL you want to read the response from, given that subsequent validation is not conducted for the ensuing requests.

Notably, the response is only readable if it constitutes a specific JSON format, namely *{"result": ...}*. However, this remains a commonly-encountered format for APIs, which significantly increases this issue's severity impact. Nevertheless, this vulnerability is limited by the minimal control permitted over the request body, which only allows the attacker to control the contents of the *params* parameter.

In addition, since the source of the bug pertains to a network fetch from an extension context, protections that would typically exist on cross-origin requests are disabled, such as mixed content (*http* or *https*) and SameSite. This grants attackers the ability to initiate dangerous CSRF requests, since network requests are sent with cookies. One can pertinently note that all cookies will be sent simultaneously even if SameSite is set, which facilitates CSRF attacks on endpoints typically protected by SameSite cookies. An attacker may also target internal endpoints on *localhost* or internal networks.

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

The fact that the code sends a POST request to the target URL by default is worthy of mention. However, this can be altered to a cross-origin GET request by utilizing a 301 or 302 redirect, which will ensure that the cookies will still be sent.

To mitigate this issue, Cure53 advises either removing the code that permits setting a custom RPC URL, or configuring the fetch middleware (*createFetchMiddlewareOriginal*) with the *redirect*[3] option set to *error*, which would prevent the middleware from performing redirects. The proposed fix should be carefully reviewed before deployment as it may incur functionality breakage if any feature utilizing the middleware relies on the usage of redirects.

BIN-04-002 FIXED WP1: Popup undismissed on client-side redirects *(Low)*

***Note**: This issue was fixed by the Trust Wallet team and the fix was verified by Cure53 via inspecting a diff. The problem as described no longer exists.*

Testing confirmed the capability to redirect the top-level page using JavaScript without dismissing the popup extension page. This behavior can be leveraged by an attacker to send requests asking users to initiate actions seemingly on the behalf of unrelated web applications.

Nonetheless, this issue is partially mitigated due to the fact that the most sensitive actions are prompted to the user in a popup displaying the origin that initiated the request. That being said, this vulnerability could be leveraged in tandem with the weakness described in ticket BIN-04-003 for enhanced security impact.

To mitigate this issue, Cure53 advises dismissing the popup each and every time a renderer-initiated navigation is conducted.

[3] https://developer.mozilla.org/en-US/docs/Web/API/fetch#:~:text=to%20follow.-,error,-%3A%20Abort

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

BIN-04-003 FIXED WP1: CSP's sandbox directive facilitates *Null* origin *(Low)*

***Note**: This issue was fixed by the Trust Wallet team and the fix was verified by Cure53 via inspecting a diff. The problem as described no longer exists.*

The observation was made that the origin displayed in the extension popup is displayed as *null* when a request is sent to the Trust Wallet from a sandboxed page via the sandbox CSP directive, or in the event a popup is opened from within a sandboxed iframe. This behavior is considered problematic because the user will not be privy to the exact page that originally requested the action.

Additionally, if users legitimately connect their wallets to a dApp that requested permission from a sandboxed top-level page, an attacker would be able to leverage the permissions granted, since they would also be able to forge a dApp with *null* origin.

This issue had previously been resolved during the prior audit, as documented under ticket BIN-03-006. However, upon further analysis, two new variations were unveiled that can be utilized to bypass the current mitigations.

To mitigate this issue, Cure53 advises retracting requests to the Trust Wallet extension from senders that contain a *null* origin.

BIN-04-004 FIXED WP1: Lack of optimal *watchAsset* validation incurs DoS *(Low)*

***Note**: This issue was fixed by the Trust Wallet team and the fix was verified by Cure53 via inspecting a diff. The problem as described no longer exists.*

Testing verified that only basic parameter checks are performed to confirm whether the parameters are empty when adding assets. As a result of this process, malicious websites are offered at least two pathways by which to exploit these parameters for DoS attacks.

Following a successful DoS attack, users may be unable to access their wallet, requiring the deletion and reinstallation of the extension to resolve the issue. This directly impacts the wallet's availability. The two scenarios in question are detailed below, pertaining to a DoS via an excessively lengthy string and a DoS via an array respectively.

DoS #1:
The first approach involves the use of an excessively lengthy string by adding an asset with a very long image URL and decimals, for instance. After the user agrees to add the asset and clicks the *Add token* button, the UI will persist in the same page until automatically logged out after a certain period of time.

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

DoS #2:
The second method involves the use of an array by adding an asset with an array as the symbol, for instance. After the user agrees to add the asset and clicks the *Add token* button, the dashboard will crash and render a blank page. The user will be permanently unable to access their wallet unless they reinstall the extension.

To mitigate this issue, Cure53 advises implementing sufficient validation on the parameters of the *watchAsset* method. For example, the type check should be implemented to ensure the parameter types are expected, rather than simply checking whether it is empty. Moreover, the length of the parameter should also be checked to guarantee the maximum range has not been exceeded.

BIN-04-005 FIXED WP1: Lack of *addEthereumChain* type validation incurs DoS *(Low)*

***Note**: This issue was fixed by the Trust Wallet team and the fix was verified by Cure53 via inspecting a diff. The problem as described no longer exists.*

Testing confirmed a lack of validation on the type of parameter when adding a new blockchain. As a result, an attacker can add a new blockchain with a native currency that uses an array as its symbol. After the blockchain is added, the wallet dashboard UI will crash due to the expectation that the symbol constitutes a string.

After a successful DoS attack, the user will be permanently unable to access their wallet, requiring the deletion and reinstallation of the extension to resolve the issue. This directly impacts the wallet's availability.

To mitigate this issue, Cure53 advises implementing the check in question for all parameters - including their type - to ensure they match the expected data structure. Additionally, one can recommend verifying the string length to ensure values are unexcessive.

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

Conclusions

To finalize this report, Cure53 will now summarize the observations and findings identified during the CW14 pentest against the Trust Wallet Browser Extension. In relation to the Trust Wallet UI and associated security indicators, a previously fixed issue had resurfaced due to an unaccounted variation. This behavior purports sandboxing a page via the sandbox CSP directive, whereby an attacker could spoof the origin indicator in the extension popup and render a *null* origin display. This issue also affected an alternative code path related to the permissions granted by users to specific dApps; in the event the user grants permission to a legitimate sandboxed dApp just once, an attacker would be able to impersonate said permissions. Additional guidance on this flaw is offered in ticket BIN-04-003.

Similarly, a bypass for a *High*-severity issue that had already been previously fixed was uncovered. Here, Cure53 noted that an attacker can utilize a malicious custom RPC URL to bypass the current validations in place and thus partially read the response of cross-origin pages by abusing the *eth_sendRawTransaction* method (see BIN-04-001).

Another pertinent finding was discovered whilst analyzing the interaction between the web page and the extension popup. This specifically pertains to an inconsistency whereby the popup page is not dismissed when the triggering page is navigated away, which may confuse users and potentially facilitate the ability to instigate phishing attacks (see BIN-04-002).

Generally speaking, the code base was subjected to rigorous review and compromise techniques, which verified strong efficacy in minimizing the attack surface. The limited volume of identified issues clearly indicates the development team's successful integration of abundant precautionary measures to enhance the security of the Trust Wallet Browser Extension. Ample evidence suggests that the DApps Platform team is not only aware of common security errors but has taken proactive steps to mitigate and prevent them.

Despite the highly constrained attack surface, the audit team evaluated every single accessible method via *window.trustwallet.request* in an attempt to uncover any risk inducing behaviors. With regards to data validation, Cure53 was pleased to acknowledge that basic checks have been implemented on the data following the previous testing round, including ensuring that the *rpcUrl* is valid and the data is not empty.

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

Nevertheless, the observation was made that the program fails to perform supplementary checks on the data type and content, thereby enabling unexpected parameter types to be passed into the extension. This, in turn, will initiate UI crashes and ultimately allow for DoS-related attacks, as outlined in tickets BIN-04-004 and BIN-04-005.

Even though the code is developed using TypeScript, which performs type checking during static compilation, Cure53 highly advises conducting checks on the incoming parameters during dynamic code execution. This would verify both the type and content to ensure they do not exceed the maximum assigned values.

In conclusion, Cure53 believes that an exemplary security standard can be achieved by addressing and mitigating all findings documented in this report, though the DApps Platform team has evidently already constructed an admirable foundation upon which to integrate future development augmentations and elevate security proficiency.

Cure53 would like to thank Eve Lam and Yeferson Licet from the DApps Platform Inc. team for their excellent project coordination, support, and assistance, both before and during this assignment.