Fine penetration tests for fine websites

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

Pentest-Report BlockWallet Browser Addon 06.2022

Cure53, Dr.-Ing. M. Heiderich, BSc. T.-C. Hong, MSc. R. Peraglie, M. Kinugawa

Index

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

Introduction

"BlockWallet is the most private, non-custodial browser extension wallet, where users can store funds and interact with their favourite blockchain applications privately."

From https://blockwallet.io/

This report - entitled BLW-01 - details the scope, results, and conclusory summaries of a source-code-assisted penetration test and source code audit against the BlockWallet browser addon, plus additional specific features and areas of interest. The work was requested by Virtual Privacy OU in April 2022 and initiated by Cure53 in late May and early June 2022, namely in CW21 and CW22. A total of fourteen days were invested to reach the coverage expected for this project.

The testing conducted for BLW-01 was divided into two separate work packages (WPs) for execution efficiency, as follows:

- **WP1**: Source-code-assisted penetration tests against BlockWallet browser addon
- **WP2**: Source-code-assisted deep-dives against specific features and areas of interest

Cure53 was granted access to applications, sources, assistful documentation, test-user accounts, as well as any alternative means of access required to complete the audit. For these purposes, the methodology chosen was white-box, and a team of four senior testers was assigned to the project's preparation, execution and finalization. All preparatory actions were completed in May 2022, namely in CW20, to ensure that the testing phase could proceed without hindrance or delay.

Communications were facilitated via a dedicated, shared Slack channel deployed to combine the workspaces of Virtual Privacy OU and Cure53, thereby allowing an optimal collaborative working environment to flourish. All participatory personnel from both parties were invited to partake throughout the test preparations and discussions.

One can denote that communications proceeded smoothly on the whole. The scope was well-prepared and clear, no noteworthy roadblocks were encountered throughout testing, and cross-team queries were kept to a minimum as a result. Virtual Privacy OU delivered excellent test preparation and assisted the Cure53 team in every respect to procure maximum coverage and depth levels for this exercise. Cure53 gave frequent status updates concerning the test and any related findings, whilst simultaneously offering prompt queries and receiving efficient, effective answers from the maintainers.

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

Live reporting was not requested, which in hindsight proved a sufficient decision considering the relatively low severity levels of the findings detected.

Regarding the findings in particular, the Cure53 team achieved comprehensive coverage over the WP1 and WP2 scope items, identifying a total of eight. Two of these findings were categorized as security vulnerabilities, whilst the remaining six were deemed general weaknesses with lower exploitation potential. Generally speaking, the overall yield of findings could be considered relatively minimal for a scope of this magnitude and complexity, which naturally constitutes a positive indication of the platform's security strength.

Following the completion of the audit, the testing team was able to confirm that the addon already exhibits a solid security posture, which is corroborated by the fact that the majority of all findings were merely considered general weaknesses and should be trivially easy to address and mitigate.

Furthermore, that the maximum severity rating assigned to any finding in this report constitutes *Medium* only strengthens this viewpoint. This lack of any significant attack surface remains a favorable conclusory outcome and the Virtual Privacy OU team certainly deserves all plaudits for their efforts in providing such a stable addon from a security perspective. Nevertheless, some leeway for improvement certainly persists. Cure53 recommends heeding all guidance offered in this report to elevate the platform to first-rate status.

The report will now shed more light on the scope and testing setup as well as provide a comprehensive breakdown of the available materials. Subsequently, the report will list all findings identified in chronological order, starting with the detected vulnerabilities and followed by the general weaknesses unearthed. Each finding will be accompanied by a technical description and Proof of Concepts (PoCs) where applicable, plus any relevant mitigatory or preventative advice to action.

In summation, the report will finalize with a conclusion in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the BlockWallet browser addon - plus additional specific features and areas of interest - giving high-level hardening advice where applicable.

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

Scope

- **Source-code-assisted penetration tests and audits against BlockWallet browser addon**
  - **WP1**: Source-code-assisted penetration tests against BlockWallet browser addon
    - extension monorepo:
      - https://github.com/block-wallet/extension/tree/073e7c631ce81a259d28bdf860cbeb2260913145
    - extension-background submodule:
      - https://github.com/block-wallet/extension-background/tree/c90b265f43fef685e67781fc01e91cba77b56d9b
    - extension-provider submodule:
      - https://github.com/block-wallet/extension-provider/tree/02b78a2845c3df820b327f733f99ce30827b70b4
    - extension-ui submodule:
      - https://github.com/block-wallet/extension-ui/tree/acd024b041904693aa1290f8d19a54eb3fb515c0
    - All sources were shared
  - **WP2**: Source-code-assisted deep dives against specific features and areas of interest
    - Hardware Wallet support (Trezor and Ledger)
      - https://github.com/block-wallet/eth-trezor-keyring
      - https://github.com/block-wallet/eth-ledger-bridge-keyring/
      - https://github.com/block-wallet/eth-ledger-bridge-keyring/tree/gh-pages
    - Implementation of Keyring, signatures, etc.
    - Implementation of injected provider
    - Communication between dApp, provider, UI and background
    - Tornado Cash feature (privacy pools)
  - **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 sections list all vulnerabilities and implementation issues identified throughout the testing period. Please note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Furthermore, each vulnerability is given a unique identifier (e.g., *BLW-01-001*) to facilitate any future follow-up correspondence.

BLW-01-003 WP2: Tornado relayer reveals wallet provider *(Medium)*

***Note**: This issue has been addressed by the BlockWallet team and the measures were successfully verified by Cure53. Support for Tornado was removed entirely.*

Testing confirmed that the same Tornado relayer from BlockWallet is always utilized when withdrawing from the privacy pool. This substantially reduces the anonymity set, since outsiders can simply observe the *rewardAccount* transaction history and deduce that any addresses that have interacted with it use BlockWallet and/or constitute related addresses. Further assessment also revealed that the BlockWallet Tornado relayer is exclusively used by BlockWallet users since it does not appear on the relayer list on the Tornado Cash website, thereby further reducing user privacy and increasing susceptibility to associated attacks.

Affected file:
extension-background-c90b265f43fef685e67781fc01e91cba77b56d9b/src/controllers/blank-deposit/tornado/config/relayers.ts

Affected code:

```
const relayers: { [network in AvailableNetworks]: string } = {
    goerli: 'goerli-relayer.blockwallet.io',
    mainnet: 'mainnet-relayer.blockwallet.io',
    bsc: 'bsc-relayer.blockwallet.io',
    polygon: 'polygon-relayer.blockwallet.io',
    arbitrum: '',
    avalanchec: '',
    optimism: '',
    xdai: '',
};
```

Configurations from https://mainnet-relayer.blockwallet.io/v1/status:

```
{"rewardAccount":"0x2209Efc366594f813e1FcDd9401560A69f1575f4"
```

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

Transaction history from https://etherscan.io/address/0x2209Efc366594f813e1FcDd9401560A69f1575f4#internaltx:

Parent Txn Hash	Block	Age	From	To	Value
0x02ee380d1c9aedcc51…	14826782	2 days 18 hrs ago	Tornado.Cash: 0.1 ETH	0x2209efc366594f813e1…	0.01662306023285 Ether
0xa23de1ce958422e3a8…	14826642	2 days 18 hrs ago	Tornado.Cash: 0.1 ETH	0x2209efc366594f813e1…	0.02151285237745 Ether
0x4e9f13a3a99488e86ff…	14825302	3 days 1 min ago	Tornado.Cash: 0.1 ETH	0x2209efc366594f813e1…	0.01075248042655 Ether
0x7f62e6d9809c97eb1c…	14802533	6 days 16 hrs ago	Tornado.Cash: 1 ETH	0x2209efc366594f813e1…	0.0270128429819 Ether
0x503a2f88b644fd409a1…	14746383	15 days 15 hrs ago	Tornado.Cash: 0.1 ETH	0x2209efc366594f813e1…	0.023712831489 Ether
0xa0105c2c2dd9c10825…	14731647	18 days 26 mins ago	Tornado.Cash: 1 ETH	0x2209efc366594f813e1…	0.01889250937425 Ether
0x75481fbd201a76ebd7…	14731642	18 days 28 mins ago	Tornado.Cash: 1 ETH	0x2209efc366594f813e1…	0.01876461432235 Ether
0xfa06f3b463092212c78…	14722833	19 days 10 hrs ago	Tornado.Cash: 1 ETH	0x2209efc366594f813e1…	0.02383818895375 Ether
0x1b74b3df1bfca0fc9c5e…	14706194	22 days 1 hr ago	Tornado.Cash: 10 ETH	0x2209efc366594f813e1…	0.05534915161155 Ether

Fig.: rewardAccount *transaction history*.

Two methods can be applied to mitigate this issue. Firstly, one could utilize a random relayer when withdrawing from the privacy pool. Secondly, the BlockWallet Tornado relayer can be registered as a public relayer[1] to increase the anonymity set, since typical users remain able to use it.

BLW-01-005 WP1: DoS via improper property access on token symbol *(Medium)*

***Note**: This issue has been fixed by the BlockWallet team and the fix was successfully verified by Cure53. The problem as described no longer exists.*

The discovery was made that one can completely “brick” the BlockWallet browser addon when importing a custom token with a malicious symbol attached. As a result, users will lose access to their wallet if a backup has not been made since the effect persists even following a wallet reset.

Steps to reproduce:

1. Switch network to Rinkeby.
2. Add token with address *0xaE2b9d47a9125dEE92650d8DF93FeF9A3E62d8aD*
3. Observe that a *__proto__* token is imported.
4. Note that the addon will subsequently be rendered nonfunctional.

Affected file:
extension-ui-acd024b041904693aa1290f8d19a54eb3fb515c0/src/components/AssetsList.tsx

[1] https://docs.tornado.cash/general/how-to-become-a-relayer

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

Affected code:

```
<span className="text-xs text-gray-600">
       {formatCurrency(
              toCurrencyAmount(
                     asset.balance || BigNumber.from(0),
                     state.exchangeRates[asset.token.symbol],
                     asset.token.decimals
              ),
       {
              currency: state.nativeCurrency,
              locale_info: state.localeInfo,
              showSymbol: true,
       }
)}
</span>
```

The underlying issue here pertains to the fact that one can access unintended JavaScript internal objects. When the code attempts to access *state.exchangeRates[asset.token.symbol]* with the expectation that it will return a number, *state.exchangeRates["__proto__"]* actually returns an internal object and an exception will be thrown in this scenario. Similar names - including *constructor* and *toString* - also persist the same behavior.

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

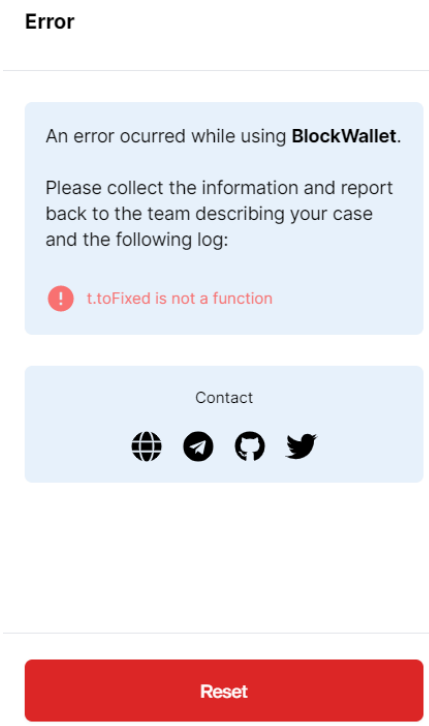

Fig.: Thrown exception when malicious token added.

Rather than utilize an object and key-value pairs, Cure53 advises leveraging an array to store and reference the token symbol. Alternatively, the *Object#hasOwnProperty* method can be used to check if a property exists on the object rather than the prototype chain.

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 lead to an exploit but might 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 way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

BLW-01-001 WP2: Low-gas Flashbots transactions may leak to mempool *(Low)*

***Note**: This issue has been fixed by the BlockWallet team and the fix was successfully verified by Cure53. The problem as described no longer exists.*

Testing confirmed that BlockWallet offers the option to send a transaction via Flashbots RPC to prevent it from being broadcasted to the public mempool. However, one should note the following stipulation from the official Flashbots documentation: *"Transactions under 42,000 gas, such as simple ether transfers, are rejected by the Flashbots relay. As a result, we will forward these to the public mempool instead."*[2]

To mitigate this issue, Cure53 advises displaying a relevant warning when a transaction's estimated gas is lower than 42,000.

BLW-01-002 WP1: Account-export dialog popup persists after defocus *(Info)*

***Note**: This issue has been fixed by the BlockWallet team and the fix was successfully verified by Cure53. The problem as described no longer exists.*

The discovery was made that the account-export dialog popup containing the wallet private key does not automatically close after a defocus. Since the dialog contains sensitive data such as the private key, an oblivious user may simply defocus the dialog after exporting it. A malicious attacker with access to the user's machine will then be able to steal the private key by simply re-opening the addon popup.

Steps to reproduce:

1. Navigate to *Accounts* > *Export* and export the private key.
2. "Blur" (defocus) the addon popup.
3. Open the addon again and observe that the private key remains displayed.

To mitigate this issue, one can recommend simply navigating back to the main page of the popup once the user has defocused it. This process would draw a parallel with the associated Metamask behavior.

[2] https://docs.flashbots.net/flashbots-protect/rpc/quick-start/#key-considerations

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

BLW-01-004 WP1: Certain transaction dialogs lack network indicator *(Info)*

Testing confirmed that the network indicator displaying the current network is not present for most transaction types. This may result in situations whereby users send transactions within an incorrect network, in the eventuality they had previously and obliviously altered the current network.

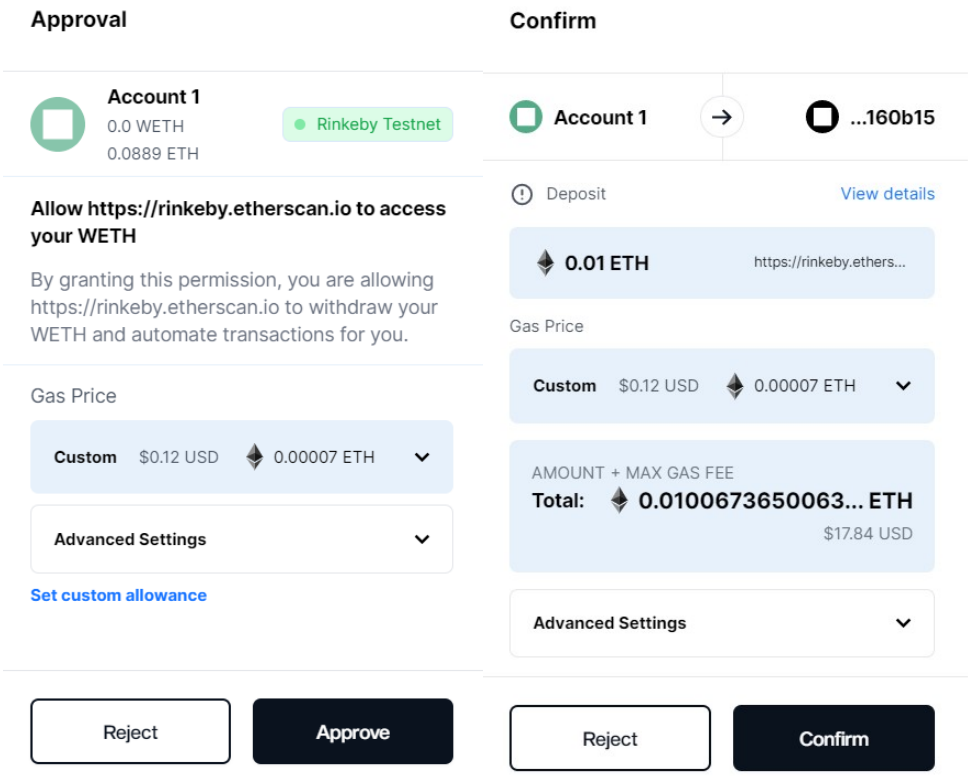

Fig:. (left) ERC-20 approval tx with network indicator (top right); (right) Typical tx without.

To mitigate this issue, Cure53 advises including the network indicator regardless of the transaction type.

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

BLW-01-006 WP1: *loglevel localStorage* stored in every origin *(Info)*

The discovery was made that the BlockWallet browser addon stores the *localStorage* entitled *loglevel* in every origin accessed by users. Since the websites may use the *localStorage* of the same name, the value overwritten by the addon can cause hitherto unforeseen behaviors such as denial of service. The aforementioned issue can be reproduced via the following steps.

Steps to reproduce:

1. Set BlockWallet to *Default Browser Wallet*.
2. Navigate to any website.
3. Open DevTools.
4. Execute *localStorage.loglevel* on the console.
5. Observe that a "WARN" string will be returned.

If these values are deemed necessary, one can recommend storing them in the addon's storage rather than the website's *localStorage* to mitigate this issue.

BLW-01-007 WP1: Improper hostname check in *isCompatible* function *(Info)*

***Note**: This issue has been fixed by the BlockWallet team and the fix was successfully verified by Cure53. The problem as described no longer exists.*

The discovery was made that a non-strict host name check is conducted in the *isCompatible* function defined within the BlockWallet browser addon. The function compares the current hostname with hostnames hardcoded in the array and returns a boolean value. However, in this comparison, since the *String#includes()* method[3] determining whether one string may be found within another string is used, unexpected hosts can control the return value. For example, the check returns "*false*" not only for *opensea.io* but also for the *cure53-opensea.io* and *opensea.io.cure53.de* domains, which do not originate from the *opensea.io* domain.

The affected code was found in the following excerpt and highlighted next.

Affected file:
https://github.com/block-wallet/block-extension-provider/blob/6a7d87e4264aa7f1207b9865b4f07aa43808017c/src/utils/site.ts#L8

[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/includes

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

Affected code:

```
export const isCompatible = (): boolean => {
    for (let i = 0; i < incompatibleSites.length; i++) {
        if (window.location.hostname.includes(incompatibleSites[i])) {
            return false;
        }
    }
```

At the time of testing, the boolean value returned by this function is simply stored in a variable and remains unused for any processing. Therefore, this behavior does not incur any tangible or significant issues at present. However, this may change in the eventuality that the value is used for important processing in future builds.

To prevent unexpected hosts from controlling the comparison processing, Cure53 advises strictly checking the hostname on every occasion. In addition, note that currently this function does not check if the protocol constitutes *https:*. Depending on how this function will be utilized in the future, the permitted *http:* URLs may facilitate vulnerabilities via Man-in-the-Middle attacks.

BLW-01-008 OOS: CSS injection via *connect.trezor.io postMessage* *(Low)*

***Note**: This issue has been fixed by the Trezor team and the fix was successfully verified by Cure53. The problem as described no longer exists.*

Whilst investigating Trezor's connection flow, a CSS injection issue was discovered in the *connect.trezor.io* domain.

Affected page:
https://connect.trezor.io/8/webusb.html

In the page offered above, a *postMessage* listener is set whilst the CSS specified in the *postMessage* is applied to the button element placed in the page. However, since this process lacks a message sender origin check, arbitrary origins can apply arbitrary CSS to the button element by sending the *postMessage*. This behavior does not constitute an issue related to the BlockWallet browser addon specifically; however, since the addon navigates to the affected domain when connecting Trezor, the issue remains pertinent. To provide an example, any would-be attacker could abuse this bug to successfully instigate phishing attacks against BlockWallet users.

The issue can be reproduced by opening the following HTML and clicking the "go" button. If the PoC functions as intended, a QR code and a message prompting users to install the fake mobile application will be displayed on the Trezor domain as a result of the CSS injection.

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

PoC:

```
<button onclick="go()">go</button>
<script>
function go(){
    window.open(`javascript:setInterval(function()
{opener.postMessage({"style":"{\\"background\\":\\"url(//vulnerabledoma.in/
phishing.png)\\",\\"background-size\\":\\"100%\\",\\"pointer-
events\\":\\"none\\",\\"background-repeat\\":\\"no-
repeat\\",\\"position\\":\\"absolute\\",\\"top\\":\\"0\\",\\"left\\":\\"0\\",\\"
width\\":\\"100%\\",\\"height\\":\\"100%\\"}"},'*')},1000);document.write('Pleas
e check another window')`,"_blank");
    location="https://connect.trezor.io/8/webusb.html";
}
</script>
```

The affected code was found in the following file. As can be deduced, the process lacks an essential origin check.

Affected file:
https://connect.trezor.io/8/js/webusb.0fbc64462b57921010f4.js

Affected code:

```
c = function() {
    //No sender's origin check
    var t = n(i().mark((function t(e) {
        var r, o, s, a, c, f;
        return i().wrap((function(t) {
            for (; ; )
                switch (t.prev = t.next) {
                [...]
                case 8:
                    [...]
                    c = document.createElement("button"),
                    "string" == typeof r.style ? (f = JSON.parse(r.style),
                    Object.keys(f).forEach((function(t) {
                        Object.prototype.hasOwnProperty.call(c.style, t) &&
                          c.style.setProperty(t, f[t])
                    }
                    ))) : c.className = "default",
                    [...]
                    ))),
                    document.body && document.body.append(c);
                case 14:
                case "end":
                    return t.stop()
                }
        }
```

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

```
        ), t)
    }
)));
```

To mitigate this issue, Cure53 strongly recommends reporting the error to the vendor as soon as possible and requesting an urgent fix. The vendor should validate the sender's origin and accept messages sent from trusted origins only.

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

Conclusions

The impressions gained during this report - which details and extrapolates on all findings identified during the CW21 and CW22 testing against the BlockWallet browser addon plus additional specific features and areas of interest by the Cure53 team - will now be discussed at length. To summarize, the confirmation can be made that the components under scrutiny have garnered a positive impression.

The testing scope of the security engagement was divided into two distinct categories: general browser-addon security and specific application features.

Regarding browser addon security, the Cure53 team initiated testing to determine whether the permissions given to the addon are not overly permissive; the URL matching rules for content scripts are not too aggressive; and that no sensitive files are exposed via *web_accessible_resources*. Subsequently, the testing team investigated any potential that the addon could introduce additional issues and erroneous behaviors to the websites via CSP bypass, for example. Positively, no issues were detected in this regard, with the exception of a potential issue related to *localStorage* as documented in ticket BLW-01-006.

Elsewhere, the potential for XSS was evaluated in considerable depth. Owing to the usage of React and lack of *dangerouslySetInnerHTML*, the testing team was able to confirm that the addon is unlikely to be vulnerable to XSS.

The communication between BlockWallet's page, content, and background script was also extensively evaluated. Generally speaking, the confirmation was made that the communication offers an exceptionally minimal attack surface via malicious external dApp applications.

Concerning the specific features of an Ethereum wallet as browser addon, assessments were conducted to determine whether the injected provider correctly and securely implemented a plethora of standard procedures. Toward this, the display for EIP-20, EIP-712, EIP-721, and EIP-1102 were deemed to offer the correct data, whilst for EIP-3085 it only allows a set of known EVM compatible networks to be added and rejects otherwise, which is considered a strong security practice.

The security of the UI provided evidence of phishing attack awareness with an explicit intention to provide solid and integer UI transitions. However, the testing team detected that the account-export data dialog does not reset the page after a defocus, which could allow malicious attackers with temporary access to instigate private-theft as documented in ticket BLW-01-002. In addition, the discovery was made that the network indicator is

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

not displayed for some transaction types, which may facilitate transactions sent via different networks; see ticket BLW-01-004 for further details. Moreover, testing confirmed that transactions that are ineligible for Flashbots relay are still sent to the RPC, which could render them leakable to the mempool (see BLW-01-001).

Whilst examining the addon logic, a DoS via improper property access was discovered. If one were to insert a custom token with a crafted name, internal JavaScript objects will be referenced, thereby causing thrown exceptions and the addon to become bricked (see BLW-01-005). The integration of Tornado Cash was also subject to deep-dive investigation. Here, the confirmation was made that the inherent privacy is substantially reduced due to the continued usage of fixed relayers, as documented in ticket BLW-01-003.

Additionally, the flow of the hardware wallet connection was assessed by the testing team. Although the addon side was deemed vulnerability-free, a CSS injection issue was detected via the Trezor origin utilized in the connection process (see BLW-01-008). This behavior should be communicated to the vendor at the earliest possible convenience to ensure that a sufficient fix can be deployed for all Trezor users.

Finally, the confirmation was made that BlockWallet does not attempt to reimplement cryptography but alternatively leverages well-tested and widely-used cryptographic libraries in order to sign Ethereum transactions and generate the zkSNARK proofs relayed to the Tornado contracts.

In summary, Cure53 is happy to conclude that the BlockWallet browser addon has garnered a strong impression following the completion of this audit. Though the addon is undoubtedly production-ready and all specific features adhere to BlockWallet's claims regarding privacy and security standards, a multitude of general mitigations and best-practice implementations can be integrated to raise the security posture to an exemplary status, as evidenced by the conclusory findings of this report.

Cure53 would like to thank Iman Hossini from the Virtual Privacy OU team for his excellent project coordination, support and assistance, both before and during this assignment.