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

Pentest-Report Towo Bifrost Wallet & API 06.2021

Cure53, Dr.-Ing. M. Heiderich, MSc. F. Fäßler, MSc. R. Peraglie

Index

Introduction

"Towo Labs develops secure crypto wallets and decentralized applications. We're also running public blockchain infrastructure."

From https://towolabs.com/

This report describes the results of a security assessment carried out by Cure53 and focused on the security posture of the Towo Bifrost complex. The project, which entailed both source-code assisted penetration tests and dedicated auditing efforts, specifically investigated the Towo Labs Bifrost crypto wallet application, together with backend API endpoints connected to the app.

To give some context, the work was requested by Towo Labs AB in late March 2021 and then scheduled for early June 2021. Cure53 executed the work in a punctual manner, looking at the Towo Labs Bifrost scope in CW23. A total of ten days were invested to reach the coverage expected for this project by a team of three senior testers. These were selected on the basis of the matching skills and expertise and assigned to this project's preparation, execution and finalization.

In order to make sure all key aspects are covered and adequately tracked, the work was split into two separate work packages (WPs):

- **WP1**: Code-assisted Penetration-Tests against Bifrost Crypto Wallet App & API
- **WP2**: Code-assisted Deep-Dives against specific Features & Areas of Interest

As can be deduced from the code access, the methodology chosen here was white-box. Given this premise, Cure53 was given access to all relevant sources in scope, precompiled binaries, test-supporting documentation and everything else that was useful for acquiring good coverage levels and depth.

All preparations were done in late May and early June, namely in CW22, so that Cure53 could have a smooth start. Communications during the test were done using a dedicated and shared Slack channel, which combined workspaces of Towo Labs and Cure53. All participating personnel could join the channel and take part in the test-relevant discussions. Communications were very smooth and not many questions had to be asked since the scope was well-prepared and clear. No noteworthy roadblocks were encountered during the test. It should be added that the Slack channel was also used to outline further focus areas set forward by Towo Labs for this examination. These were:

- Security properties of the Error Handling Rollbar
- Secure handling of Recovery Phrase Generation/Import/Storage
- General properties around App Protection (PIN/Biometrics)
- Security posture of the Dependency Tree
- General checks against Risky Practices

The project moved forward efficiently. Cure53 sent frequent status updates about the test's progress and emerging findings. Live-reporting was neither requested nor deemed necessary. The Cure53 team managed to get very good coverage over the WP1-2 scope items and made only one security-relevant discovery. This flaw was classified as a general weakness with very low exploitation potential. No actual vulnerabilities were spotted, which is a fantastic result for the Towo Wallet App and the connected backend parts. It is obvious that the Towo Labs team has incorporated very good security practices from very early on and that the attack surface is kept minimal and well-protected.

The report will now shed more light on the scope and test setup as well as the available material for testing. This will be followed by a chapter covering the test methodology and the areas investigated by the Cure53 team, even if no issues stemmed from a given approach. This will help avoid duplicate work in-house and can give Towo Labs clarity regarding the test coverage and depth.

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

Next, the spotted issue will be presented. The report will then close with a conclusion in which Cure53 will elaborate on the general impressions gained throughout this test and reiterate the verdict about the perceived security posture of the Towo Labs Bifrost Crypto Wallet App & connected backend API endpoints.

Scope

- **Code-assisted Penetration-Tests & Audits against Bifrost Crypto Wallet App**
    - **WP1**: Code-assisted Penetration-Tests against Bifrost Crypto Wallet App & API
        - Wallet App Repository
            - https://github.com/TowoLabs/bifrost-wallet
        - Backend API Repository
            - https://github.com/TowoLabs/bifrost-backend
    - **WP2**: Code-assisted Deep-Dives against specific Features & Areas of Interest
        - Particular attention was given to the wallet's backend API, verifying that all communication occurs over HTTPS (to the Bifrost Wallet backend only) and that a compromised backend cannot affect the user's recovery phrase or transaction signing process
        - Further in focus was improper platform usage, insecure data storage, insecure communication or insufficient cryptography.
        - Dependencies interacting with the user's recovery phrase, such as wallet generation and transaction signing, were also covered.
    - **Binaries for several platforms were shared with Cure53**
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

Test Methodology

This section documents the testing methodology applied during this engagement and sheds light on the various areas of the codebase set for inspection and audit. It further clarifies which areas were examined by Cure53 but did not yield any findings.

WP1: Penetration tests against Bifrost crypto wallet app & API

- In the first work package, Cure53 audited the Bifrost crypto wallet for Android and iOS, as well as the backend implementing the API. The first section will cover tests conducted against the mobile apps, and the second one describes testing against the backend API.
- With the source code of the application provided, a local development build was created to enable a more in-depth audit.
- The network communication between the application and the API backend was tested by attempting a MitM attack on a physical, Android phone having the app installed. It was found that the app does not disable any SSL security checks.
- Frida[1] was set up in order to allow easier inspection of networking requests issued by the mobile application. Direct attacks against the API were conducted with the Burp proxy.
- While the application only connects to a trusted API endpoint, for the sake of the audit it was assumed that the backend API could be compromised. In that case, the user's crypto funds should be safe. The mobile app logic was audited to ensure that a malicious backend could not trick the user via Phishing, sending a transaction to a wrong address or through other ways of exposing critical data to the server.
- The backend was found to determine the gas price and gas limit for a transaction. This could give a compromised backend the ability to supply extremely high fees, potentially profiting from processing the block in a node. However, the app obtains confirmation from the user transparently and a user should notice the unnecessary high fees. To improve this, the app could implement a sanity check to prevent sending transactions if the fees are higher than the actual transaction value.
- Another potential attack for a compromised backend concerns sending of file paths for cryptocurrency icons, which could be risky. Yet the mobile app properly verifies the paths and no path traversal or inclusion of external resources could be found.
- Further, the backend was inspected and showed a good posture in typing requests and responses. Generally, the usage of TypeScript has been a very good choice.
- The backend was found to act as a client for multiple blockchains, mostly relaying signed transactions and account balances to the appropriate chain, whilst processing no personal information. Therefore, no authentication and no authorization seem to be required. This could be used by malicious actors to use the backend as an anonymous proxy for transactions, but this also cannot be fully prevented. Verifying some request

[1] https://frida.re/

headers or even attaching a client secret, which could be changed with each release, would at least require some efforts for abuse.

- Another typical attack vector against mobile applications is the ability to deep-link content or even send malicious intents in case of Android. Such functionality often leads to PIN bypasses due to directly linking to views that should only be accessible after an *authorization* step. However, no deep-linking or custom intent handlers are being used.
- The newly embedded browser feature became available during the test. Embedded web views inside of applications, especially those that expose functionality to the loaded site, can be a critical component.
- A major focus for the audit was the communication between JavaScript running within the WebView and the App. It was tested if a malicious site could use the RPC mechanism to gain code execution or perform other dangerous actions within the native app.
- The RPC also injects JavaScript into the page to return results. This could potentially lead to UXSS issues, thus the implementation was reviewed. To sanitize the response, *JSON.stringify* is used. This function allows the injection of *__proto__ fields,* which could be risky, but no issue could be found. Other forms of JavaScript injections are generally prevented.
- Another important part to audit were the encryption layer and the native storage. Bifrost is using AES in CTR mode, which lacks authentication, however in the context of this app using secure native storage, it is not believed to be any issue.
- The application was inspected for common Cross-Site Scripting pitfalls like *dangerouslySetInnerHTML, injectJavaScript* of which none were found to be exploitable.
- Additionally, no custom underlying database could be identified that made use of any query language, rendering the search of associated vulnerabilities redundant.

WP2: Deep-dives against specific features & areas of interest

This section describes the deep dives that were performed on dependencies and specific areas of concern listed by Bifrost.

- **Error Handling Rollbar**
  - The wallet app uses the Rollbar SDK to collect information about errors and crashes from the users' devices. During testing, no leaks of critical information were observed. However, to address the concern that these errors could leak a user's masterkey or other critical information, it is recommended to configure the data scrubbing feature of Rollbar. By doing so, a granular programmatically scrubber will be able to remove potentially sensitive data from information sent to Rollbar.
- **App Protection (PIN/Biometrics)**
  - Biometrics were found to be outsourced to the *react-native-fingerprint-scanner* dependency. The library is used as intended without further customizations. It also catches authentication errors appropriately.

    - During dynamic testing of the exposed intents and activities, it was found that the biometrics dependency inserts another activity into the Android Manifest during the build process. It was verified that calling this activity directly does not circumvent the PIN entry.
    - Further, no way was found to bypass the PIN confirmation or extract the user's masterkey without knowledge of the PIN or valid biometrics. This also holds for the user's masterkey and derived private keys.
    - It could be considered to additionally encrypt the user's masterkey and derived private keys with the PIN/password of the user and only unlock the keys for direct usage. This has two benefits: A) It is almost impossible for the app to forget asking the user for the PIN on transactions as it is a mandatory requirement. B) Given a PIN/password with sufficient entropy is used, attackers will not be able to extract the user's masterkey or perform transactions from an unlocked and rooted device that was physically obtained.
    - In this course, it is advisable to lift the 6-digit PIN-length constraint and allow users to opt-in to stronger security by choosing an arbitrary password, with arbitrary characters and length.
- **Dependency Tree**
    - The mobile app uses several dependencies as wrappers for native features of iOS and Android. The *react-native-webview* dependency was audited, as it was being used by the new embedded browser feature. It was checked if they used safe settings for the native web view components.
    - To implement the platform-specific storage features, the *react-native-encrypted-storage* dependency is used. It was reviewed if the implementation benefitted from safe settings for both operating systems.
    - Bifrost makes use of the *react-native-svg* library to load remote icons of individual chains as SVGs from the backend. SVGs are prone to Cross-Site Scripting vulnerabilities if they are handled badly in the app and the backend is compromised. While this is currently not the case, it is advised that those images are added statically to the app to mitigate this risk.
    - The *react-native-fingerprint-scanner* component is used for the Biometrics as described in the *PIN/Biometrics* section. It can generally be said that those dependencies are third-party repositories that are always susceptible to prime threats like repository hijacking or framework vulnerabilities.
- **Risky Practices**
    - The complex destroys the wallet of a user after entering the incorrect PIN ten times to aggressively prevent wallet-access. Although this is a desirable security mechanism, it could be used by attackers to destroy financial assets from a user's phone that does not have a backup of their recovery phrase.
    - Alternatively, instead of completely destroying the attacked wallet, an exponential backoff time could be introduced. For instance, if the time to wait was doubled after

each failed attempt, attackers cannot deny access longer than they need to enter the PINs repeatedly. This prevents users from accidentally locking themselves out of their own wallets. At the same time, it achieves the state wherein a malicious attacker cannot try thousands of PINs automatically. Though it should be noted that accurate time tracking can be difficult as well because an attacker could change the system time.

- The RPC handler is passing user-controlled data as an attribute name to the console object *console[rpcMessage.level]()*. This is generally risky, but it is mitigated by the Wallet code to conform to an enum value.

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals 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.

TOW-01-001 WP1: Missing TLS certificate pinning for the backend API (*Info*)

It was found that the app does not include a certificate or a fingerprint that will be trusted when connecting to the Bifrost Wallet backend. This introduces the risk of the Bifrost wallet accepting any TLS certificate on the first-connection attempt. As a consequence, there is a chance for the connection being intercepted by attackers that have access to a generally trusted certificate authority. This could be abused to read and intercept all traffic between the Bifrost backend and the intercepted users. As the backend must not be trusted, this issue is purely *Informational.*

Affected file:
android/app/src/main/res/xml/network_security_config.xml

Affected code:
```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
       <domain-config cleartextTrafficPermitted="true">
       <domain includeSubdomains="true">10.0.2.2</domain>
       <domain includeSubdomains="true">localhost</domain>
       </domain-config>
</network-security-config>
```

It is recommended to implement strict certificate pinning by supplying the certificate or a cryptographic fingerprint of the certificate within the app. This could be done by configuring the *network_security_config.xml* file on Android[2] and the *Info.plist* file on iOS[3]. The certificate or its issuing CA certificate should be solely trusted when connecting to the backend API. This prevents attackers from intercepting traffic against users that initially connect to the Bifrost wallet.

Developer Note: *“This network_security_config.xml is for development. We will be using a separate config for production, including certificate pinning. We also use HSTS, DNSSEC, CAA for all of our domains. Additionally, we have ordered a registry lock for our domains.”*

[2] Cert Pinning, Android https://developer.android.com/training/articles/security-config#CustomTrust
[3] Cert Pinning, iOS https://developer.apple.com/news/?id=g9ejcf8y

Conclusions

As can be seen by the near complete lack of findings, the Towo Bifrost Wallet has a very strong security standing. After spending ten days testing the scope in June 2021, three members of the Cure53 can clearly praise the security-related decisions taken by Towo Labs during the development phase so far. While the app is still in active development, and many critical features have not been fully implemented yet, this test leaves a very good first impression.

Through the communication with the client before and during the test, it is clear that the developers have a strong sense for security and understand the responsibility that comes with their product and code. TypeScript is used to implement the backend and mobile app, which is a great choice. Proper typing is generally translating to positive outcomes in terms of security and it does appear to be the case here, too.

The code is clean and easy to understand, which makes it easy to audit. Three main threat models were audited. Those were direct attacks against the backend server, attacks against the mobile apps from malicious applications on the phone or done by physical attackers, as well as attacks against the apps from a compromised backend.

Even with a fully compromised backend, the financial assets and personal data needs to remain as safe as can be. This stresses the value of the principle of least privilege that was found throughout the architecture. This could especially be observed when the impact of TOW-01-001 was nullified. The attack-surface of the mobile apps was minimized with best efforts, particularly thanks to utilizing well-tested dependencies correctly and without security-relevant customizations that open room for error.

Concluding from the results and impressions gathered during this audit, it can be said that the Bifrost architecture is on its optimal path to furnish a very secure crypto wallet.

Cure53 would like to thank Markus Alvila and Patrik Sletmo from the Towo Labs AB team for their excellent project coordination, support and assistance, both before and during this assignment.