Fine penetration tests for fine websites

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

Audit-Report BOB MetaMask Snap Codebase & Build 02.2024

Cure53, Dr.-Ing. M. Heiderich, Dr. N. Kobeissi

Index

- Introduction
- Scope
- Test Methodology
  - Scope Definition
  - Methodology
- Identified Vulnerabilities
  - BOB-01-001 WP1: Instance of outdated cryptography (Low)
- Conclusions

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

Introduction

"BOB (Build on Bitcoin) is the first Bitcoin L2 with full EVM compatibility & native Bitcoin support empowering everyone to build and innovate on Bitcoin. BOB (Build on Bitcoin) enables DeFi and innovation across all fields of Bitcoin use cases & experimentation. Whatever you're building on Bitcoin, BOB is your swiss-army-knife for all things build on Bitcoin."

From https://www.gobob.xyz/

This report describes the results of a penetration test and source code audit against the BOB MetaMask Snap and its codebase. The work was requested by Distributed Crafts Ltd. in January 2024 and performed by Cure53 in February 2024, namely in CW06 and CW07. A total of five days were invested to achieve the expected coverage for this project. The work was divided into two separate work packages (WPs). These were as follows:

- **WP1**: Source code audits against BOB MetaMask Wallet Snap & codebase
- **WP2**: Snap & Chain specific feature reviews against BOB MetaMask Wallet Snap

Cure53 was provided with sources and all other access necessary to perform the tests, using a white box methodology. A team of two senior testers was assigned to prepare, execute and close this project.

All preparations were made in late January and early February 2024, namely CW05, so that Cure53 could have a smooth start. Communication during the test was done via a dedicated shared telegram channel between the development team and Cure53, to which all involved personnel from both parties were invited. Communication was smooth and there were not many questions to be asked, the scope was well prepared and clear, and there were no significant roadblocks during the test.

Cure53 maintained an open line of communication with the project team via Telegram (the client's chosen communication platform), ensuring that any clarifications or additional information required for the audit could be addressed promptly. Regular updates and preliminary findings were shared with the project team to facilitate immediate feedback or corrective action. Cure53 provided frequent status updates on the audit and related findings; live reporting was not specifically requested for this audit.

The Cure53 team was able to achieve good coverage of the WP1-WP2 scope items and was able to identify only one finding, which was classified as a security vulnerability. This audit and review of the BOB MetaMask Snap, specifically its Bitcoin ordinals functionality, revealed an overall positive impression. While there are areas for improvement, particularly in cryptographic practices, the design and implementation of the Snap strongly emphasizes security principles. Testing of the Bitcoin ordinals confirmed careful consideration for secure transaction handling and innovative use of blockchain technology.

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

The report now goes into more detail about the scope and test setup, as well as the material available for testing. This section is followed by a chapter detailing the testing methodology used in this exercise. This is to show the audience which areas of the software in scope were covered and which tests were performed even though only one issue was found.

The report will then list all findings in chronological order, first the vulnerabilities found and then the general weaknesses discovered during this test. Each vulnerability is accompanied by a technical description, a PoC where possible, and mitigation or fix advice. The report will then end with a conclusion in which Cure53 will elaborate on the general impressions gained throughout this test and share some words about the perceived security posture of the scope that is the BOB MetaMask Snap and its codebase.

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

Scope

- **Pen.-tests & code audits against BOB MetaMask Snap codebase & builds**
  - **WP1**: Source code audits against BOB MetaMask Wallet Snap & codebase
    - **Primary focus:**
      - General tests & attacks against Browser Add-OnsExtension Snap-Ins, independently of the specific use case as a crypto wallet snap.
    - **Sources:**
      - https://github.com/bob-collective/btcsnap/tree/master
    - **Commit in Scope:**
      - *ad88a73fe378321173fbc9574357bf76b322a2eb*
  - **WP2**: Snap- & Chain-specific feature reviews against BOB MetaMask Wallet Snap
    - **Primary focus:**
      - Specific features, such as but not exclusive to the following: features enabling users to trade & inscribe inscriptions (Ordinal, BRC20, etc.)
    - **Sources:**
      - *See above*
  - **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

This section of the audit report outlines the comprehensive methodology applied by Cure53 during the penetration test and source code review of the BOB MetaMask Snap Codebase & Build. The audit was conducted over approximately five days and focused on identifying vulnerabilities, security flaws, and areas for improvement within the codebase.

Scope Definition

Prior to commencing the audit, Cure53 conducted thorough preparation activities, including familiarization with the BOB MetaMask Snap environment, code base, and underlying technologies. This preparation included reviewing documentation provided by the customer, setting up the test environment, and defining clear audit objectives. The scope of the audit covered the entire codebase of the BOB MetaMask Wallet Snap, with particular attention to:

- Source code audits focused on general tests and attacks against browser add-on/extension snap-ins.
- Review of snap and chain specific features, especially those that allow users to trade & write inscriptions (Ordinal, BRC20, etc.).

Methodology

A team of two senior security testers performed a line-by-line manual review of the codebase, focusing on critical areas identified during the automated scan phase and known security-sensitive components. This included a thorough review of:

- Cryptographic implementations and usage patterns: the audit team evaluated the cryptographic implementations and usage patterns present in the code, with the goal of uncovering any misuse of cryptographic algorithms or practices that could compromise the security of user data and transactions.
- Authentication and authorization mechanisms: This examination was critical to verifying that the Snap employed robust authentication measures to prevent unauthorized access and that its authorization protocols were correctly implemented.
- Data validation and remediation practices: Data validation and sanitization practices within the Snap were also rigorously evaluated. This part of the review was critical to confirm that the Snap effectively neutralizes input-based threats by validating, sanitizing, and securely handling user-supplied data. In the context of this Snap, this largely meant the secure use of MetaMask Snap API calls.
- Dependency analysis, especially for third-party libraries with known vulnerabilities: The audit team performed a dependency analysis with a special focus on third-party libraries integrated into the Snap.

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

This review identified only one finding related to outdated cryptographic practices (BOB-01-001). Although this finding was rated as low severity, it's still significant because it affects overall security. The issue involved the use of weak password hashing (low iteration counts in PBKDF2) and missing integrity checks in cryptographic functions. Cure53 recommends adopting more robust cryptographic approaches to address these concerns.

Special attention was given to the unique features of the snap, such as trading and inscriptions. Testers evaluated the implementation for security vulnerabilities in the feature-specific logic and potential misuse scenarios.

Throughout the testing process, findings were documented with a clear description and recommendations for mitigation. This documentation served as the basis for the audit report.

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 found during the testing period. Notably, the findings are listed in chronological order rather than by severity, with the severity rank in parentheses following the title heading for each vulnerability. In addition, all issues are assigned a unique identifier (e.g., *BOB-01-001*) to facilitate future follow-up correspondence.

BOB-01-001 WP1: Instance of outdated cryptography *(Low)*

Client Note: *Distributed Crafts acknowledges the finding. The code is coming from the original MetaMask Zion Snap and covers the Lightning Network (LN) integration. We decided to leave the code in but not promote usage of the LN functionality for regular users. However, we see potential usage of the code for developers or others that might want to experiment with the BOB Snap.*

It was found that the BOB MetaMask Snap uses cryptographic constructions to secure user data that do not strictly adhere to best practices. While the intent to secure data is clear, there are several aspects that make this implementation outdated based on modern cryptographic standards:

- Low number of iterations in PBKDF2: The code uses PBKDF2 with 1,000 iterations to derive the encryption key from the private key and salt. The iteration count is considered low by current standards, although given that the input is a private key and the output is trusted to be the output of a cryptographically secure pseudorandom number generator, the impact is potentially limited.
- Lack of data integrity checks: The code does not appear to implement any error handling for encryption and decryption, or any integrity checks (such as using an AEAD or HAMC) to ensure that the ciphertext has not been corrupted or tampered with.
- Use of a non-native cryptography library: While CryptoJS was an important library in the era before native encryption support was implemented in browsers via the Web Crypto API and similar offerings, relying on it today exposes cryptographic pitfalls to many common problems with JavaScript runtime cryptography, such as the potential for side-channel attacks.

Affected files:

- *rpc/saveLNDataToSnap.ts*
- *rpc/getLNDataFromSnap.ts*

Affected code:

```
export async function saveLNDataToSnap(
  domain: string,
  snap: Snap,
  walletId: string,
```

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

```
  credential: string,
  password: string,
) {
  const privateKey = (await getHDNode(snap, LNHdPath)).privateKey.toString(
    'hex',
  );
  const salt = CryptoJs.lib.WordArray.random(16);
  const key = CryptoJs.PBKDF2(privateKey, salt, {
    keySize: 16,
    iterations: 1000,
  });

  const iv = CryptoJs.lib.WordArray.random(16);
  const encrypted = CryptoJs.AES.encrypt(credential, key, { iv: iv });
  const encryptText = salt.toString() + iv.toString() +
encrypted.toString();
  const result = await getPersistedData(snap, 'lightning', {});

[...]

export async function getLNDataFromSnap(
  domain: string,
  snap: Snap,
  {
    key,
    walletId,
    type = 'get',
  }: GetLNDataFromSnap
): Promise<string>
 [...]
 const encryptText = lightning[walletId].credential;
 const salt = CryptoJs.enc.Hex.parse(encryptText.substring(0, 32));
 const iv = CryptoJs.enc.Hex.parse(encryptText.substring(32, 64));
 const encrypted = encryptText.substring(64);
 const privateKey = (
   await getHDNode(snap, LNHdPath)
 ).privateKey.toString('hex');
 const key = CryptoJs.PBKDF2(privateKey, salt, {
   keySize: 512 / 32,
   iterations: 1000,
 });
 const credential = CryptoJs.AES.decrypt(encrypted, key, { iv: iv });
```

The following recommendations can be followed to address the issues described above:

- Switch to a different password hash function or increase the number of iterations in PBKDF2 (optional): Given that the input is a salted pseudorandom private key, using PBKDF2 with a low number of iterations may potentially be okay. However, it is

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

currently recommended to use PBKDF2 with at least 600,000 iterations, or to switch to a more secure password hash function, such as Scrypt.

- Workaround for integrity checking: Implement an Authenticated Encryption with Associated Data (AEAD) scheme, such as AES-GCM or ChaCha20-Poly1305. AEAD schemes provide both confidentiality and integrity protection for the data. When using a non-AEAD encryption scheme, add a Hash-Based Message Authentication Code (HMAC) to ensure the integrity and authenticity of the data. This step involves generating an HMAC over the ciphertext using a separate key (derived from the original key or independent) and verifying the HMAC before decryption.
- Error Handling Remediation: Implement comprehensive error handling around cryptographic operations to catch and properly handle errors such as incorrect keys, corrupted data, or decryption failures. This includes logging the errors in a secure manner for analysis without exposing sensitive information.
- Migrate to native cryptography libraries: Migrate from CryptoJS to the Web Crypto API or other native cryptographic libraries provided by the platform. The Web Crypto API is designed to be secure against many common vulnerabilities in JavaScript cryptography, including side-channel attacks. It's also optimized for performance and supported by modern browsers:
  - Perform a feasibility analysis for integrating the Web Crypto API into your application, considering factors such as browser support and the specific cryptographic operations required.
  - Replace CryptoJS cryptographic calls with their counterparts in the Web Crypto API, ensuring that all cryptographic operations are performed using this more secure and efficient interface.
  - Thoroughly test the updated implementation to ensure that it works correctly in all supported environments and that the migration has not introduced any new issues.

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

Conclusions

Cure53's audit of the BOB MetaMask Snap codebase covered a wide range of components, including a full source code review and feature-specific testing.

A significant but low severity finding from the audit was the identification of outdated cryptographic practices within the BOB MetaMask Snap, as detailed in BOB-01-001. These included the use of low iteration counts in PBKDF2 and a lack of integrity checks in cryptographic constructions. Cure53 included recommendations for the use of more robust cryptographic constructions.

Throughout the audit process, Cure53 maintained effective communication with the project team, fostering a collaborative environment. This approach ensured that any ambiguities or issues could be addressed promptly, allowing for real-time feedback and adjustments to the testing strategy as needed.

The responsiveness and commitment of the BOB MetaMask Snap project team played a critical role in the success of the audit. Their willingness to provide the necessary documentation, access, and clarification facilitated a thorough and efficient audit process, underscoring the importance of collaboration between security auditors and project teams in identifying and mitigating security risks.

The audit of the BOB MetaMask Snap, with a focus on its Bitcoin ordinal offering, left a very positive impression regarding its security posture. Despite identifying areas for improvement, particularly in cryptographic practices, the Snap's design and implementation demonstrate a strong commitment to security principles. The feature-specific testing of bitcoin ordinals highlighted the careful consideration given to secure transaction handling and the innovative use of blockchain technology within the Snap.

Cure53 would like to thank Zhixi, Gregory Hill and Dominik Harz from the Distributed Crafts Ltd. team for their excellent project coordination, support and assistance both before and during this assignment.