QuillAudits

Audit Report
April, 2023

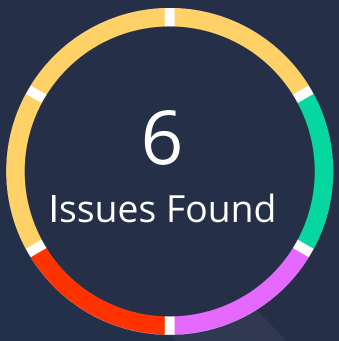
For

BEYOND IMAGINATION TECHNOLOGIES

Table of Content

Executive Summary

Project Name	BIT-Wallet
Overview	The BIT-Wallet is a blockchain project that is making a blockchain wallet for users (Currently only on Polygon Testnet and Mainnet).
Timeline	29 November, 2022 - 24 January, 2023
Scope of Audit	The scope of this pentest was to analyze the source code AND corresponding wallet for quality, security, and correctness. **In Scope:** Source-Code *https://github.com/etherneel/BIT-Wallet*
Fixed In	*https://github.com/etherneel/BIT-Wallet/commits/main*



🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	1	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	3	0	1

Techniques and Methods

Throughout the pentest of BIT-Wallet, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Wireshark, etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest

- Burp Suite
- Frida
- Nmap
- Metasploit
- Horusec
- Postman
- Netcat
- Nessus and many more.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Types of Severities

High
A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium
The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low
Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational
These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Checked Vulnerabilities

- ✓ Improper Platform Usage
- ✓ Insecure Data Storage
- ✓ Insecure Communication
- ✓ Insecure Authentication
- ✓ Insufficient Cryptography
- ✓ Insecure Authorization
- ✓ Client Code Quality
- ✓ Code Tampering
- ✓ Reverse Engineering
- ✓ Extraneous Functionality
- ✓ Indirect Object ReferenceFunctionality Abuse
- ✓ Business Logic
- ✓ Wallet Seed Phrase Protection
- ✓ Previous attack exploits
- ✓ Transfer of tokens from and in application
- ✓ Rate Limit Issues and Bruteforcing

Manual Testing

High Severity Issues

1. CRYPTOJSSECRET Along with other keys Leked

CRYPTOJSSECRET key is being used to encode and decode the address, private key, and mnemonic for application usage.
These type of keys should not be leaked in source code on github and such platforms as it can lead to attackers stealing data from victims in an encrypted format and decrypting it using such secret keys.

APIKEYCOVLANT Key should also not be directly accessible like this as it has rate limit of 5 requests per second. If the attacker gets such keys then they can exhaust the limit and make it necessary for you to buy premium plans which can lead to harm for business.

Vulnerable File Location: src/utils/index.js

Recommendation

- Remove such keys from GitHub repo before opening it to the public.
- Encrypt such keys with salt and keep salt hidden to avoid any such issue.

Status

Resolved

Medium Severity Issues

2. Multiple Deprecated Libraries in Package-Lock.json

Package-lock.json Stores files that can be useful for the dependency of the application. This is used for locking the dependency with the installed version. It
will install the exact latest version of that package in your application and save it in package. This arises a problem if the dependency used has an exploit in the version mentioned. It can create a backdoor for an attacker.

Vulnerable Dependencies and location:

Loader-utils	package-lock.json
nth-check	package-lock.json
minimatch	package-lock.json
terser	package-lock.json
got	package-lock.json
decode-uri-component	package-lock.json

Recommendation

- Update all the above Mentioned Dependencies
- Remove any Library Not needed.

Impact

Multiple of these libraries have public exploits and CVE-registered issues that have been patched and can help your application stay more secure from any dependency-vulnerable issues.

Status

Resolved

3. Self-Transfer With no Warning

Self-Transfer shouldn't take place in normal conditions and sometimes users tend to copy an address to send crypto but some users can mistakenly select their own address and can just execute a self-transfer. When such a transaction takes place there should be a popup or a warning sign to show the user that a self-transfer is taking place and the user will just receive funds to themselves.

Steps to Reproduce:

1. Login into the wallet
2. Copy your own address
3. Click on send (Can be testnet or mainnet)
4. Paste the copied address
5. Click on Send

POC:- TXN

Recommendation

Issue a warning when copied address is the same as the wallet address.

Impact

Loss of user funds due to just paying gas fees.

Status

Resolved

4. Use of math.random()

The JavaScript Math.random() function is designed to return a floating point value between 0 and 1. It is widely known (or at least should be) that the output is not cryptographically secure. When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.
As the Math.random() function relies on a weak pseudorandom number generator, this function should not be used for security-critical applications or for protecting sensitive data. In such a context, a cryptographically strong pseudorandom number generator (CSPRNG) should be used instead.

Vulnerable location: src/screens/Welcome/components/Step7.js

Recommendation

Use strong generation methods as Crypto.getRandomValues() when required.

Reference: *Stakoverflow*

Status

Resolved

Low Severity Issues

5. Clickjacking

Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website.

Recommendation

Use the X-Frame-Options to prevent (or limit) pages from being embedded in iFrames.

Status

Acknowledged

Informational Issues

6. Terms of services are ambiguous

In crypto space users usually tend to read terms to know what info the application is collecting and storing. In your terms and service, a particular statement can be interpreted incorrectly and make you a loss of users.

Statement — Information provided by you when creating an account;

As, while creating account we get mnemonic phrase for our account this statement can be interpreted as that the user's phrases are being stored on your end even if they are not. Kindly specify such statement along with "THAT YOUR PRIVATE KEY AND BACKUP SEED PHRASE MUST BE KEPT SECRET AT ALL TIMES"

Status

Resolved

Closing Summary

In this report, we have considered the security of the BIT-Wallet. We performed our audit according to the procedure described above.

Some issues of High, low and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Dapp audit is not a security warranty, investment advice, or an endorsement of the BIT-Wallet Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multi-step process. One audit cannot be considered enough. We recommend that the BIT-Wallet Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2023

For

BEYOND
IMAGINATION
TECHNOLOGIES

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com