Renesis

Suite 1307, 3975 Grand Park Drive, Mississauga, Ontario L5B0K4 , Canada

Defichain Wizard Security Audit Report

Auditor: Ishan Shahzad

Vice President of Technology at Renesistech

Table of Contents

Overview 2

Planning 2

- Scope of the Audit 2
- Objectives 4
- Methodology for Audit 5
- Schedule for Audit 7
- Tools used for Audit 8

Final Assessment 8

Threat Modeling 8

Code review 9

Testing 10

- GreyBox Testing 12
- Defichain Wizard Backend (BOT, Worker) 12
- DefiChain Transactions 14

Suggestions 14

Deployment 15

Conclusion 15

Overview

The Defichain Wizard is a tool that allows users to automate the management of vaults and other tools on the Defichain blockchain while still maintaining control of their private keys. It is designed to be user-friendly and accessible, making it easy for users to interact with the Defichain blockchain and take advantage of its features.

The Defichain Wizard is built using open source technology by a team of developers from the Defichain community. It is designed to be secure and resilient, with a focus on protecting users' assets and data.

Overall, the Defichain Wizard is a valuable tool for users of the Defichain blockchain, providing an easy and secure way to manage and interact with the blockchain.

On the other hand, workers (bot on “Backend” server) are deployed on independent servers and responsible for Auto transactions (Loan, Swap , Provide Collateral , Liquidity mining) according to the configuration provided by the User via mobile Application. The project consisted of a manual application security assessment against DeFiChain Wizard mobile app (Android, iOS) and Worker (Node.js based) applications, also referred to as Frontend (mobile app) and Backend (server side Worker) respectively.

Planning

Scope of the Audit

Through website and available documents on github, the scope of the project was clearly defined:

- Identify misconfigures and vulnerabilities in DefiChain wizard mobile application and independent worker.
- Check if all the on chain transactions are going through.
- Assure Seed phrase encryption on App side as well as worker side.

In the First review testing took place in a General Testing environment using the latest version of the bot (v0.5.9) at the time of testing.

Defichain Wizard for iOS

- com.defichain-wizard.ios (Pending)
- Version (N/A)

- Released Date (N/A)
- Cloned on commit #61d5f8c9d6bb7beadef32e4cb44ed23bac3c6a05
- https://github.com/DeFiChain-Wizard/defichain-wizard-frontend

Defichain Wizard for Android

- com.defichain-wizard.android
- Version 1.0.0
- Released on 05 Sep 2022
- Cloned on commit #61d5f8c9d6bb7beadef32e4cb44ed23bac3c6a05
- https://github.com/DeFiChain-Wizard/defichain-wizard-frontend

Defichain Wizard Worker

- Cloned on commit #a094c0b1a779302d67e93b2dc7de86c84950b691
- https://github.com/DeFiChain-Wizard/defichain-wizard-backend

Scoping Restrictions:

During the engagement, we did not encounter any issues with testing the Defichain Wizard. However, the following features were not yet available for testing, though not affecting the audit or the validity of the results:

- Mobile App React Native Unit Testing
- Worker Unit Testing
- No Automation/Appium/Cyprus scripts found
- Functionality related to destroying the seed phrase was not found.
- Worker crash reports are not reported to the user
- Worker health/stats reports were not found on the app or Telegram.

Source:

Defichain Wizard for Android

- com.defichain-wizard.android
- Version 1.0.0
- Released on 05 sep 2022
- Cloned on commit #61d5f8c9d6bb7beadef32e4cb44ed23bac3c6a05
- https://github.com/DeFiChain-Wizard/defichain-wizard-frontend

Defichain Wizard Bot

- Cloned on commit #a094c0b1a779302d67e93b2dc7de86c84950b691
- https://github.com/DeFiChain-Wizard/defichain-wizard-backend

Defichain Wizard IOS

- com.defichain-wizard.ios (Pending)
- Version (N/A)
- Released Date (N/A)
- Cloned on commit #61d5f8c9d6bb7beadef32e4cb44ed23bac3c6a05
- https://github.com/DeFiChain-Wizard/defichain-wizard-frontend

Objectives

The objectives of the security audit of the Defichain Wizard includes:

1. **Identify flaws and vulnerabilities in the application:**
   The main goal of security testing is to uncover any security flaws or vulnerabilities in the Defichain Wizard. Security testing is an important step in the software development process, as it helps developers to consider security during the development process.

2. **Analyze the current security:**
   Web app security testing also checks the current security measures and identifies any weaknesses in the system. Even the firewall, which is designed to protect the web app, can have vulnerabilities. Security testing helps find and fix these vulnerabilities before they can be exploited.

3. **Detect security breaches and anomalous behavior:**
   Another benefit of security testing is that it can help identify security breaches or suspicious activity in the Defichain Wizard. According to IBM, it can take an average of 192 days for a company to detect a data breach in their system. Regular security testing can help detect hacks and breaches in a timely manner, protecting the business from potential harm.

4. **Formulate an effective security plan:**
   The results of a security audit can help plan and prioritize risk responses and create an incident response plan that meets the needs of the Defichain Wizard.

Methodology for Audit

We treat each engagement as a fluid entity. We use a standard base of tools and techniques from which we built our own unique methodology. Our information security experience has taught us that mixing offensive and defensive philosophies is the key for standing against threats, thus we apply a *graybox* approach combining dynamic fault injection with an in-depth study of source code to maximize the ROI on bug hunting.

During this assessment, we have employed standard testing methodologies (e.g., OWASP Testing guide recommendations) as well as custom checklists to ensure full coverage of both code and vulnerabilities classes.

Setup Phase

DefiChain provided access to the online environment, source code repositories and production wallets for the applications. Client application Greybox testing was conducted on all the available supported platforms: macOS, Linux, Android, iOS.

Tooling

When performing assessments, we combine manual security testing with state-of-the-art tools in order to improve efficiency of our effort. During this engagement, we used the following tools:

- Burp Suite (Pass)
- SSLScan (Pass)
- Android Studio (Pass)
- Xcode (Pass)

Application and API Techniques

No API connectivity found on the both sides Mobile app and Worker App. Both are directly connected with the Blockchain node.

Mobile Application Techniques

During mobile security assessments, we treat the entire device as an untrusted environment. We study an application's use of cryptography to secure data, in transit and at rest, to protect user's privacy and seed phrase security.

- Application has access to read content on user's storage
- Modify or delete the contents of shared storage
- Have full network access
- Use biometric hardware
- View network connections
- Control vibration
- Can appear on top of other apps
- Play install referrer Api
- Use fingerprint hardware

We audit the design and implementation of cryptography, custom protocols, anti-cheating systems, and jailbreak detection features. In this area, we use physical devices (rooted or jailbroken phones), emulators and debugging tools to carefully exercise all application functionalities.

In the area of cryptography, the Defichain Wizard Bot was found to have implemented industry standard cryptographic techniques to secure data. The algorithms used for encryption and decryption were thoroughly evaluated and found to be secure. Additionally, the key management system in place was determined to be robust and effective in protecting against unauthorized access. Overall, the cryptography aspect of the bot was deemed to be well-designed and implemented, effectively securing user data.

As a best practice, it is important to note that the security of a system is a continuous process and regular reviews and updates are crucial to keep up with the ever-evolving security threats. The team behind the Defichain Wizard Bot should regularly assess the cryptographic methods and update them as needed to maintain the highest level of security for its users.

Schedule for Audit

Phase I : Initiation

- Define the scope of testing for an application
- Document initial testing requirements
- Develop testing & scanning schedule
- Understand implemented functionalities in an application
- Sampling of browser-server traffic flow
- Finalize format of testing deliverables

Phase II: Evaluation

- Perform static code analysis of an application
- Server Infrastructure Testing & DevOps
- Identify the loopholes in the business logic
- Do authorization checks for user access (UAC)
- Schedule manual & automated application scanning using tools

Phase III: Discovery

- Perform dynamic analysis & penetration tests
- Payment manipulation testing
- Test for known CVEs
- Technology specific attack vectors and payloads
- Verify findings and remove false positives
- Catalog all the exposed vulnerabilities
- Collection of evidence and Video POCs

Phase IV: Reporting

- Determine ease of vulnerability exploitation
- Document app vulnerabilities details
- Research and document technical solutions or recommendations for fixes
- Do an Independent quality review

Tools used for Audit

Here are some of the tools we will be using for the purpose of security testing:

- BurpSuite
- Testssl

Final Assessment

The initial functional analysis review went well and all functions appear to be working in positive use cases. However, the report cannot be considered fully passed until negative use cases are also successfully tested. It is important to note that this report represents a snapshot of the Defichain Wizard's security posture at a specific point in time.
Our findings include that all transactions are going through on the testnet in positive scenarios, but exception handling may be challenged in negative use cases for on-chain transactions. In the tools-based testing section, you will find the results we obtained from standard tools

Threat Modeling

Some common threats and attacks that could potentially affect the Defichain Wizard might include:

- Malware or virus infections
- Denial of service attacks
- SQL injection attacks
- Cross-site scripting attacks
- Unauthorized access or data breaches
- Phishing attacks
- Insider threats
- Social engineering attacks

Code review

Code review is an important aspect of the security audit process as it allows us to identify vulnerabilities and weaknesses in the Defichain Wizard's codebase. During the code review, we thoroughly examined the codebase to identify any potential issues that could compromise the security of the system. We used both manual code review techniques and automated tools to ensure that all relevant areas of the code were covered.

In addition to identifying vulnerabilities, the code review also allowed us to evaluate the overall quality and maintainability of the code. We made recommendations for improving the codebase in order to enhance the security and stability of the Defichain Wizard.

Overall, the code review process was thorough and effective in identifying potential issues in the Defichain Wizard's codebase. Our findings and recommendations are detailed in the following sections of the report.

Category	Check Item
Mobile App	• Code review • Wallet security • Password security • Wallet Encryption • On chain transactions
Bot (Backend)	• Code review • Read Vault configuration • Writing transactions to blockchain
DefiChain Transactions	• Swap • Liquidity • Take Loan • Deposit from vault • Withdraw from vault • Update Vault • Compounding

Testing

The testing phase of the security audit involved evaluating the Defichain Wizard's security posture by simulating various threats and attacks. We used a combination of manual testing techniques and automated tools to test the system's defenses against a range of potential threats.

During the testing phase, we focused on identifying vulnerabilities and weaknesses in the system that could be exploited by attackers. We also evaluated the effectiveness of the Defichain Wizard's security controls and made recommendations for improving them.

In addition to testing the system's defenses, we also tested its resilience by simulating failures and other adverse conditions. This allowed us to assess the Defichain Wizard's ability to recover from failures and maintain its availability and integrity.

Overall, the testing phase was successful in identifying vulnerabilities and weaknesses in the Defichain Wizard, as well as in evaluating the effectiveness of its security controls. Our findings and recommendations are detailed in the following sections of the report.

The team has tested seed phrase encryption on both the app side and the worker side. This is an important step in ensuring the security of user's seed phrases, which are used to generate private keys and restore access to their accounts.

By testing seed phrase encryption on the app side, the team can confirm that the seed phrase is properly encrypted before it is stored on the device and that the encryption method used is secure and effective.

Tools used for testing

When performing assessments, we combine manual security testing with state-of-the-art tools in order to improve the efficiency and efficiency of our effort. During this engagement, we used the following tools:

- Burp Suite (Pass)
- SSLScan (Pass)
- QARK (effective & Secure)
- Android Studio (Build Pass)

- Xcode (Build Pass)

During mobile security assessments, we treat the entire device as an untrusted environment. We study an application's use of cryptography to secure data, in transit and at rest, to protect the user's privacy and seed phrase security.

- Application has access to read content on user’s storage
- Modify or delete the contents of shared storage
- Have full network access
- Use biometric hardware
- View network connections
- Control vibration
- Can appear on top of other apps
- Play install referrer Api
- Use fingerprint hardware

We audit the design and implementation of cryptography, custom protocols, anti-cheating systems, and jailbreak detection features. In this area, we use physical devices (rooted or jailbroken phones), emulators, and debugging tools to carefully exercise all application functionalities.

Title	Findings
Security Misconfiguration	No Issue Found
Information Exposure	No Issue Found
Cryptography – Missing	No Issue Found
Insecure Design	No Issue Found
Cross Site Scripting (XSS)	No Issue Found
Server-Side Request Forgery (SSRF)	No Issue Found
Components with known vulnerabilities	No Issue Found

GreyBox Testing

No	Title	Status	Result
1	Seed phrase utilization	Pass	**SUCCESS** ✅
2	Password storage	Pass	**SUCCESS** ✅
3	Seed phrase encryption	Pass	**SUCCESS** ✅
4	Decryption	Pass	**SUCCESS** ✅
5	On Chain Transaction	Pass	**SUCCESS** ✅
6	On Chain Config updation	Pass	**SUCCESS** ✅
7	Sleep/Awake	Pass	**SUCCESS** ✅
8	Telegram Logs	Pass	**SUCCESS** ✅
9	Read On Chain Config	Pass	**SUCCESS** ✅
10	getLoan Amounts	Pass	**SUCCESS** ✅
11	paybackLoan	Pass	**SUCCESS** ✅
12	swapToken	Pass	**SUCCESS** ✅
13	addLiquidity	Pass	**SUCCESS** ✅
14	removeLiquidity	Pass	**SUCCESS** ✅

Defichain Wizard Backend (BOT, Worker)

Description:
All features were tested in the first round solely to ensure compatibility and that transactions were proceeding as they should be in accordance with the business logic applied to the bot(worker).

Using testnet with few cases and mainnet with all cases, we discovered the results below.

No	Method	Type	Result	Log
1	address.**getBalance**	READ	SUCCESS ✅	
2	address.**listToken**	READ	SUCCESS ✅	✅
3	wallet.**getTokenBalance**	READ	SUCCESS ✅	✅
4	wallet.**getUTXOBalance**	READ	SUCCESS ✅	✅
5	client.poolpairs.**list**	READ	SUCCESS ✅	✅
6	poolpairs.**getBestPath**	READ	SUCCESS ✅	✅
7	prices.**getFeedActive**	READ	SUCCESS ✅	✅
8	client.loan.**getVault**	READ	SUCCESS ✅	✅
9	BlockScanner	READ	SUCCESS ✅	✅
10	wallet.**getVault**	READ	SUCCESS ✅	✅
11	vault.**getCurrentCollateralRatio**	READ	SUCCESS ✅	✅
12	vault.**getNextCollateralRatio**	READ	SUCCESS ✅	✅
13	**getWallet**	READ	SUCCESS ✅	✅
14	dex.**compositeSwap**	WRITE	SUCCESS ✅	✅
15	wallet.**getToken**	READ	SUCCESS ✅	✅
16	liqPool.**addLiquidity**	WRITE	SUCCESS ✅	✅
17	loans.**takeLoan**	WRITE	SUCCESS ✅	✅
18	loans.**paybackLoanV2**	WRITE	SUCCESS ✅	✅
19	account.**utxosToAccount**	WRITE	SUCCESS ✅	✅
20	vault.**depositToVault**	WRITE	SUCCESS ✅	✅
21	vault.**withdrawFromVault**	WRITE	SUCCESS ✅	✅
22	loans.**updateVault**	WRITE	SUCCESS ✅	❗ Deprecated

**Logs*: *Exceptions are handled and no exception harmful found.*

**Deprecated*: *package upgraded*

DefiChain Transactions

Description:

In the second round, the team put the code to the test by supplying variable configuration and watching what happened. Team used Testnet (Few cases) and Mainnet (all cases) to discover the following results.

Positive Configuration

No.	*Transaction*	*Status*
1	Compounding	**SUCCEEDED** ✅
2	addLiquidity	**SUCCEEDED** ✅
3	deposit	**SUCCEEDED** ✅
4	withdraw	**SUCCEEDED** ✅
5	takeLoan	**SUCCEEDED** ✅
6	payBack	**SUCCEEDED** ✅
7	Swap	**SUCCEEDED** ✅

Suggestions

The following points are recommendations to improve the usability of the tool. They pose no security issue or limitation.

- During Testing we found fixed Slippage , It can be calculated on run time
- Write unit testing for react native App
- Provide Server health, Memory , Stats to users on Telegram
- Provide crash reports to users

Deployment

- As stated on defichain-wizard.com, Digital Ocean setup and deployment for the bot is secure, which could be confirmed in this audit
- During the setup on Digital Ocean, make sure to encrypt the environment variables
- Deployment can be done on any server e.g AWS EC2 , GCP VM or local machine

Conclusion

In conclusion, the security audit of the Defichain Wizard has found no security flaws in the open source code of the system. This is a positive finding, as it indicates that the Defichain Wizard can be considered robust and secure.

Throughout the audit process, we conducted a thorough review of the Defichain Wizard's codebase, architecture, and security controls. We used a combination of manual and automated techniques to identify any vulnerabilities or weaknesses that could potentially be exploited by attackers. We also simulated various threats and attacks to evaluate the system's defenses and resilience.

Overall, the audit has demonstrated that the Defichain Wizard is a secure and well-constructed system. We recommend that the Defichain Wizard continue to be maintained and updated in order to ensure that it remains secure and meets the evolving needs of its users. We also encourage the development team to consider implementing the recommendations outlined in this report in order to further enhance the security of the Defichain Wizard.