Retreeb

ERC20

Smart Contract Audit Report

January 04, 2021

Introduction

1. About Retreeb

Within a rapidly changing sector, Retreeb presents a new means of payment, simple, practical, economical, which allows it to comply with the universal values such as ethics, sharing and solidarity. It targets all persons who are part of a solidarity and sustainable approach. In consideration of their adoption of the service, Retreeb commits to its users to **pay 33% of the transaction fees collected by Retreeb** to the funding of social and environmental projects. With this business model, the technical infrastructure, the redistribution of transaction fees, and the monitoring of projects, they opt for an unprecedented level of transparency in a particularly opaque sector. Concerned about environmental issues, their technological choices are determined by a desire to reduce our carbon footprint to its strict minimum. Finally, they take a new approach to payment by placing corporate social and environmental responsibility (CSR) at the heart of their ambitions.

Visit https://retreeb.io/ to know more about.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 105+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The Retreeb team has provided the following doc for the purpose of audit:

1. https://retreeb.io/assets/retreeb-white-paper.pdf
2. Short description of the intended behaviour

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Retreeb
- Token Name: Treeb
- Mainnet Address: https://ftmscan.com/token/0xc60d7067dfbc6f2caf30523a064f416a5af52963
- Languages: Solidity(Smart contract)
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck, SFuzz

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level References

Every issue in this report was assigned a severity level from the following:

Admin/Owner Privileges can be misused either intentionally or unintentionally.

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	-
Closed	-	-	-

Found During Final Audit

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Recommendations/Informational

1. **Excessive use of digits can be avoided**
   Line no - 522
   Description:
   The above-mentioned lines have a large number of digits that makes it difficult to review and reduces the readability of the code.

```
520
521     constructor(string memory _name, string memory _symbol) ERC20(_name, _symbol) {
522       _mint(_msgSender(), 1_000_000_000 * 1E18);
523     }
524   }
```

   Recommendation:
   Ether Suffix can be used to symbolize the 10^18 zeros.

2. **Unlocked Pragma statements found in the contracts**
   Explanation:
   During the code review, it was found that the contracts included unlocked pragma solidity version statements.

   It's not considered a better practice in Smart contract development to do so as it might lead to accidental deployment to a version with unfixed bugs.

   Recommendation:
   It's always recommended to lock pragma statements to a specific version while writing contracts.

Automated Audit Result

```
Compiled with solc
Number of lines: 516 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 6 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 14
Number of informational issues: 4
Number of low issues: 2
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC20

+-------+-------------+-------+--------------------+--------------+----------+
|  Name | # functions |  ERCS |     ERC20 info     | Complex code | Features |
+-------+-------------+-------+--------------------+--------------+----------+
| Treeb |      34     | ERC20 |     No Minting     |      No      |          |
|       |             |       | Approve Race Cond. |              |          |
|       |             |       |                    |              |          |
+-------+-------------+-------+--------------------+--------------+----------+
INFO:Slither:flatToken.sol analyzed (6 contracts)
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Concluding Remarks

While conducting the audits of the Retreeb smart contracts, it was observed that the contracts contain no High, Medium and Low severity issues.

The recommendations given will improve the operations of the smart contract.

- ***The Treeb contract does not contain any issues.***

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Retreeb platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes