Rocket Capital Investment

Token

Smart Contract Audit

Final Report

September 14, 2021

IMMUNE BYTES

Introduction

1. About Rocket Capital Investment

The Rocket Capital Investment Competition is a decentralized platform to source and incentivize the best in machine-learning applications for finance, ultimately leading to a decentralized autonomous fund management ecosystem driven by the community.

In this first implementation, participants stake tokens, submit their predictions and earn rewards based on their stake and on the performance of their submissions.

Visit these links to learn more about:
Company Website: https://www.rocketcapital.ai
Whitepaper: https://rocket-capital-investment.gitbook.io/rci-competition/white-paper
Smart Contract Documentation:
https://rocket-capital-investment.gitbook.io/competition-dapp/overviews/competition-overview
Web Application: https://competition.rocketcapital.ai/

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 75+ 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 RCI team has provided the following doc for the purpose of audit:

1. https://rocket-capital-investment.gitbook.io/rci-competition/

Audit Process & Methodology

The 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: Rocket Capital Investment
- Contracts Name: Child.sol, Token.sol
- Languages: Solidity(Smart contract)
- Github commit for initial audit: 26ea641b959a17b0a987f429a50d7f0fecde37ad
- Github commit for initial audit: c4ecf3f4c7875e61f70a1b147b1a7b96bb13fc99
- 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:

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	-	-	1

Contract: Token.sol

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low Severity Issues

1. **Absence of Zero Address Validation**
   Line no- 78, 85

   Explanation:
   The **Token** Contract includes a function called **authorizeCompetition**, which updates an imperative mapping, i.e., **_authorizedCompetitions**, in the contract.

```
78      function authorizeCompetition(address competitionAddress)
79      external
80      onlyAdmin
81      {
82          _authorizedCompetitions[competitionAddress] = true;
83
84          emit CompetitionAuthorized(competitionAddress);
85      }
```

   However, during the automated testing of the contract, it was found that no Zero Address validation is implemented before updating the address of the mapping.

   Although the function has already been assigned an **onlyOwner** modifier, keeping in mind the immutable nature of the smart contract, its imperative to implement input validations in function.

   Recommendation:
   A require statement should be included in such functions to ensure no invalid address is passed in the arguments.

   Amended (September 14th 2021): Issue was fixed by the **RCI** team and is no longer present in commit c4ecf3f4c7875e61f70a1b147b1a7b96bb13fc99.

Contract: ChildToken.sol

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low Severity Issues

No issues were found.

Recommendations

--

Automated Test Results

1. Token.sol

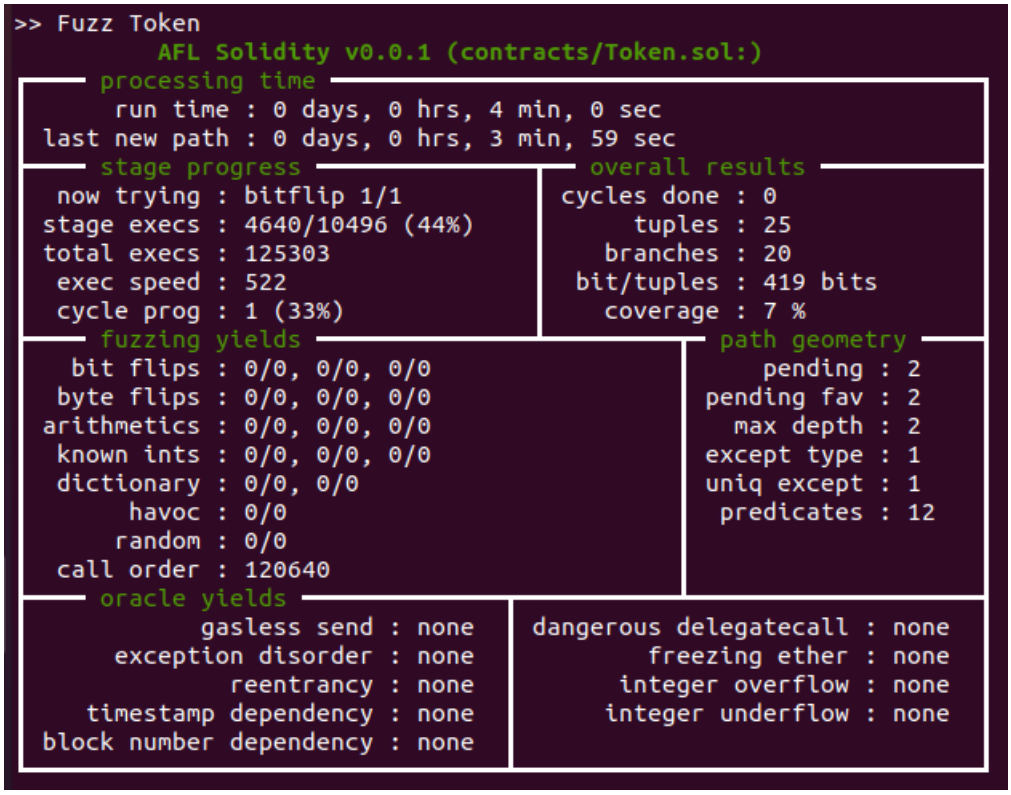
```
Compiled with solc
Number of lines: 1305 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 12 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 14
Number of informational issues: 5
Number of low issues: 3
Number of medium issues: 3
Number of high issues: 0

ERCs: ERC20, ERC165

+--------------+-------------+--------------+--------------------+--------------+----------+
|     Name     | # functions |     ERCS     |     ERC20 info     | Complex code | Features |
+--------------+-------------+--------------+--------------------+--------------+----------+
| ICompetition |      55     |              |                    |      No      |          |
|    Token     |      53     | ERC20,ERC165 |     No Minting     |      No      |          |
|              |             |              | Approve Race Cond. |              |          |
|              |             |              |                    |              |          |
+--------------+-------------+--------------+--------------------+--------------+----------+
INFO:Slither:myFlats/TokenFlat.sol analyzed (12 contracts)
```

2. ChildToken.sol

```
Compiled with solc
Number of lines: 1356 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 13 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 14
Number of informational issues: 5
Number of low issues: 6
Number of medium issues: 3
Number of high issues: 0

ERCs: ERC20, ERC165

+--------------+-------------+--------------+--------------------+--------------+----------+
|     Name     | # functions |     ERCS     |     ERC20 info     | Complex code | Features |
+--------------+-------------+--------------+--------------------+--------------+----------+
| ICompetition |      55     |              |                    |      No      |          |
|  ChildToken  |      58     | ERC20,ERC165 |     No Minting     |      No      |          |
|              |             |              | Approve Race Cond. |              |          |
|              |             |              |                    |              |          |
+--------------+-------------+--------------+--------------------+--------------+----------+
INFO:Slither:myFlats/ChildFlat.sol analyzed (13 contracts)
```

Fuzz Testing

1. **Token.sol**: -
    a. Terminal Output
    [With use of : " -g -r 0 -d 240 "]



- Excel Sheet of States for the Output of Fuzz Testing

  [With use of : " -g -r 1 -d 240 "]

  https://drive.google.com/file/d/1bTGUrxeSAyqB1yLJzvzrUdETQuDrVXIF/view?usp=sharing

Concluding Remarks

While conducting the audits of the Rocket Capital Investment smart contracts, it was observed that the contracts contained only Low severity issues. No High or Medium severity is found.

Our auditors suggest that Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

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 Rocket Capital Investment 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