4RX

FourRXToken.sol & ICOContract.sol

Smart Contract Audit Report

IMMUNE BYTES

Audits

May 21, 2021

Introduction **3**
About 4RX 3
About ImmuneBytes 3
Documentation Details **3**
Audit Process & Methodology **4**
Audit Details **4**
Audit Goals **6**
Security Level References **6**
Contract Name: ICOContract.sol **7**
Low Severity Issues 7
Informational 8
Contract Name: FourRXToken.sol **9**
Low Severity Issues 9
Informational 10
Automated Test Results **11**
Concluding Remarks **12**
Disclaimer **12**

Introduction

1. About 4RX

The purpose of the 4RX project is to build one token that reflects a wide range of crypto assets with a high potential growing curve.

4RX is the very first base price index utility token, used as an indicator that indicates the market volatility in real-time with 15-20% of total all tokens market cap valued as $360B as of April 1st 2021.

4RX indices 65 crypto assets, mostly defi projects with high potential curve for the next coming years. This enables the 4RX community to passively stake a long list of coins with one token and save a tremendous volume of money on trading and gas fees.

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 15+ 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 4RX team has provided documentation for the purpose of conducting the audit. The documents are:

1. 4RX Whitepaper.docx

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: 4RX
- Contract Name(s): *ICOContract.sol & FourRXToken.sol*
- Languages: Solidity(Smart contract)
- Github commit hash for audit:
  - **bf96812acb07dd3081e73c9a88176cc3d1a07312**
  - **852a9a8221a10623ef2adc763419857a37d33797**
- GitHub Link:
  - https://github.com/FourRX/4rx/blob/master/contracts/ICO/ICOContract.sol
  - https://github.com/FourRX/4rx/blob/master/contracts/4RX/FourRXToken.sol
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

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	-	-	6
Closed	-	-	-

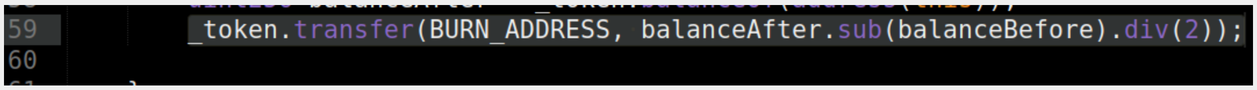
Contract Name: ICOContract.sol

Low Severity Issues

1. **Return Value of an External Call is never used Effectively**
   Line no - 43, 49, 59, 93, 102

   Description:
   The external calls made in the above-mentioned lines do return a boolean as well as other imperative values that could effectively indicate whether or not the external call was successful.

   However, the ICO contract does not use these return values effectively in some instances of the contract.



```
59            _token.transfer(BURN_ADDRESS, balanceAfter.sub(balanceBefore).div(2));
60
```

   Recommendation:
   Effective use of all the return values from external calls must be ensured within the contract.

2. **External Visibility should be preferred**

   Description:
   Those functions that are never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
   This will effectively result in Gas Optimization as well.

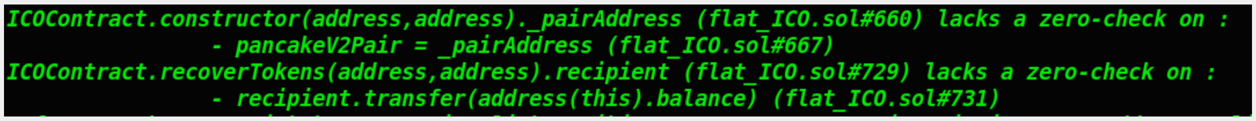
   Therefore, the following function must be marked as **external** within the contract:
   - **purchase**
   - **recoverTokens**

   Recommendation:
   If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

3. **constructor does not include Zero Address Validation**
   Line no: 250-253

   Description:
   The **constructor** initializes some of the most imperative state variables, i.e., **fourRXToken** address, **pancakeV2Pair** address in the ICO contract.

   However, during the automated testing of the contract, it was found that the constructor doesn't implement any Zero Address Validation Check to ensure that no zero address is passed while initializing this state variable.



```
ICOContract.constructor(address,address)._pairAddress (flat_ICO.sol#660) lacks a zero-check on :
                - pancakeV2Pair = _pairAddress (flat_ICO.sol#667)
ICOContract.recoverTokens(address,address).recipient (flat_ICO.sol#729) lacks a zero-check on :
                - recipient.transfer(address(this).balance) (flat_ICO.sol#731)
```

   Recommendation:
   Zero address validation should be implemented to ensure no invalid address passed as arguments.

Informational

1. NatSpec Annotations must be included

   Explanation:
   The smart contracts do not include the NatSpec annotations adequately.

   Recommendation:
   Cover by NatSpec all Contract methods.

Contract Name: FourRXToken.sol

Low Severity Issues

1. **Constant declaration should be preferred**
   Line no- 53 to 63

   Description:
   State variables that are not supposed to change throughout the contract should be declared as **constant**.

   Recommendation:
   The following state variables need to be declared as **constant**, unless the current contract design is intended.
   - **API_URL**

2. **External Visibility should be preferred**

   Description:
   Those functions that are never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
   This will effectively result in Gas Optimization as well.

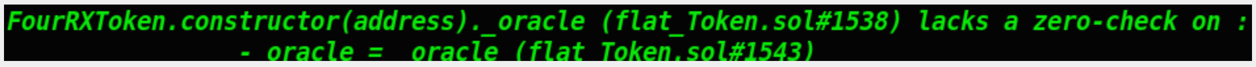
   Therefore, the following function must be marked as **external** within the contract:
   - **requestPrice**
   - **fulfill**

   Recommendation:
   If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

3. **constructor does not include Zero Address Validation**
   Line no: 250-253

   Description:
   The **constructor** initializes one of the most imperative state variables, i.e., **oracle** address in the Token contract.

However, during the automated testing of the contract, it was found that the constructor doesn’t implement any Zero Address Validation Check to ensure that no zero address is passed while initializing this state variable.



```
FourRXToken.constructor(address)._oracle (flat_Token.sol#1538) lacks a zero-check on :
                - oracle = _oracle (flat_Token.sol#1543)
```

Recommendation:
Zero address validation should be implemented to ensure no invalid address passed as arguments.

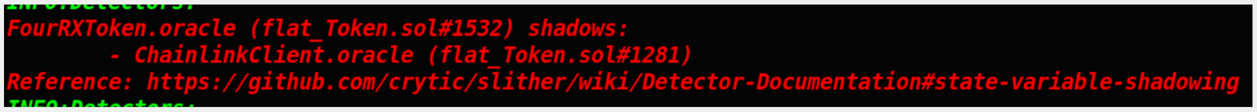
Informational

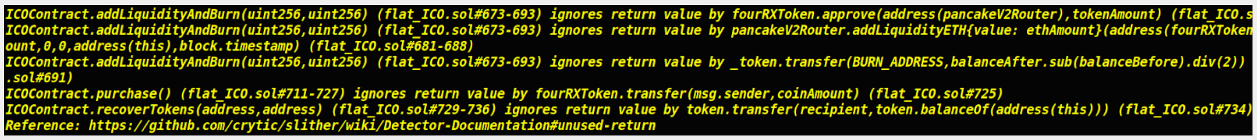
1. NatSpec Annotations must be included

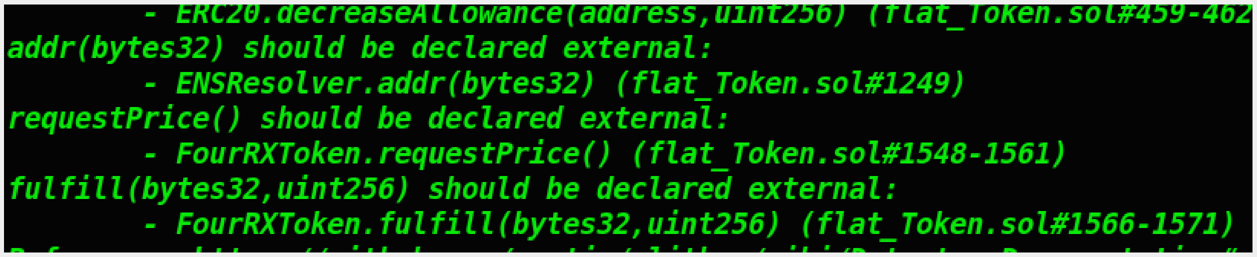
   Explanation:
   The smart contracts do not include the NatSpec annotations adequately.

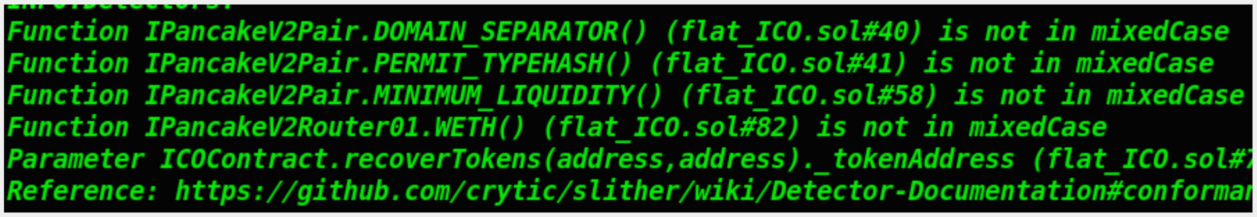
   Recommendation:
   Cover by NatSpec all Contract methods.

Automated Test Results


```
FourRXToken.oracle (flat_Token.sol#1532) shadows:
        - ChainlinkClient.oracle (flat_Token.sol#1281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
```


```
ICOContract.addLiquidityAndBurn(uint256,uint256) (flat_ICO.sol#673-693) ignores return value by fourRXToken.approve(address(pancakeV2Router),tokenAmount) (flat_ICO.s
ICOContract.addLiquidityAndBurn(uint256,uint256) (flat_ICO.sol#673-693) ignores return value by pancakeV2Router.addLiquidityETH{value: ethAmount}(address(fourRXToken
ount,0,0,address(this),block.timestamp) (flat_ICO.sol#681-688)
ICOContract.addLiquidityAndBurn(uint256,uint256) (flat_ICO.sol#673-693) ignores return value by _token.transfer(BURN_ADDRESS,balanceAfter.sub(balanceBefore).div(2))
.sol#691)
ICOContract.purchase() (flat_ICO.sol#711-727) ignores return value by fourRXToken.transfer(msg.sender,coinAmount) (flat_ICO.sol#725)
ICOContract.recoverTokens(address,address) (flat_ICO.sol#729-736) ignores return value by token.transfer(recipient,token.balanceOf(address(this))) (flat_ICO.sol#734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```


```
        - ERC20.decreaseAllowance(address,uint256) (flat_Token.sol#459-462
addr(bytes32) should be declared external:
        - ENSResolver.addr(bytes32) (flat_Token.sol#1249)
requestPrice() should be declared external:
        - FourRXToken.requestPrice() (flat_Token.sol#1548-1561)
fulfill(bytes32,uint256) should be declared external:
        - FourRXToken.fulfill(bytes32,uint256) (flat_Token.sol#1566-1571)
```


```
Function IPancakeV2Pair.DOMAIN_SEPARATOR() (flat_ICO.sol#40) is not in mixedCase
Function IPancakeV2Pair.PERMIT_TYPEHASH() (flat_ICO.sol#41) is not in mixedCase
Function IPancakeV2Pair.MINIMUM_LIQUIDITY() (flat_ICO.sol#58) is not in mixedCase
Function IPancakeV2Router01.WETH() (flat_ICO.sol#82) is not in mixedCase
Parameter ICOContract.recoverTokens(address,address)._tokenAddress (flat_ICO.sol#7
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformar
```

Concluding Remarks

While conducting the audits of 4RX smart contracts - FourRXToken.sol and ICOContract.sol, it was observed that the contracts contain only Low severity issues, along with a few areas of recommendations.

Our auditors suggest that Low severity issues should be resolved by the developers. Resolving the areas of recommendations are up to the team's discretion. 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 4RX platform or its product neither 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 Pvt Ltd.