Keep Safe Finance Smart Contract Preliminary Audit Report

Project Synopsis

Project Name	**Keep Safe Finance**
Platform	BSC, Solidity
Github Repo	Not Provided
Deployed Contract	https://bscscan.com/address/0xCAe905A3ab9304D681C73b32B6fCcBc69C9cEBb1#code
Total Duration	3 Days
Timeline of Audit	**10th September 2021 to 13th September 2021**

Contract Details

Total Contract(s)	1
Name of Contract(s)	**`CoinToken`**
Language	Solidity
Commit Hash	***Null***

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
Critical Severity	0	0
Medium Severity	0	0
Low Severity	1	0
Information	0	0
Total Found	1	0

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review, etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A. Contract Name: CoinToken

High Severity Issues
None Found

Medium Severity Issues
None Found

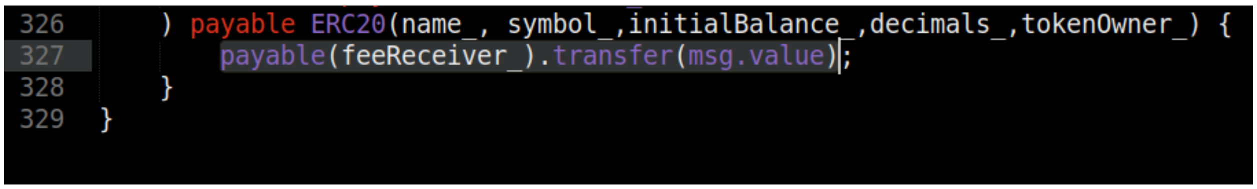
Low Severity Issues
A.1 Absence of Zero-Address Validation found in the contract
Explanation:

The CoinToken contract includes a constructor that accepts an address argument called ***feeReciever_.***

The **“*msg.value*”** amount passed while deploying the contract is transferred to this argument ***feeReciever_*** within the constructor body.

However, during the manual code review of the contract, it was found that no **zero address** input validation has been performed on this argument before initiating the transfer of **BNB.**

This might lead to an undesirable scenario where the **BNB** amount is transferred to an invalid address argument passed during the contract deployment.



```
    ) payable ERC20(name_, symbol_,initialBalance_,decimals_,tokenOwner_) {
        payable(feeReceiver_).transfer(msg.value);
    }
}
```

Recommendation:

Adequate input validations must be implemented effectively in the contract to avoid the above-mentioned scenario.

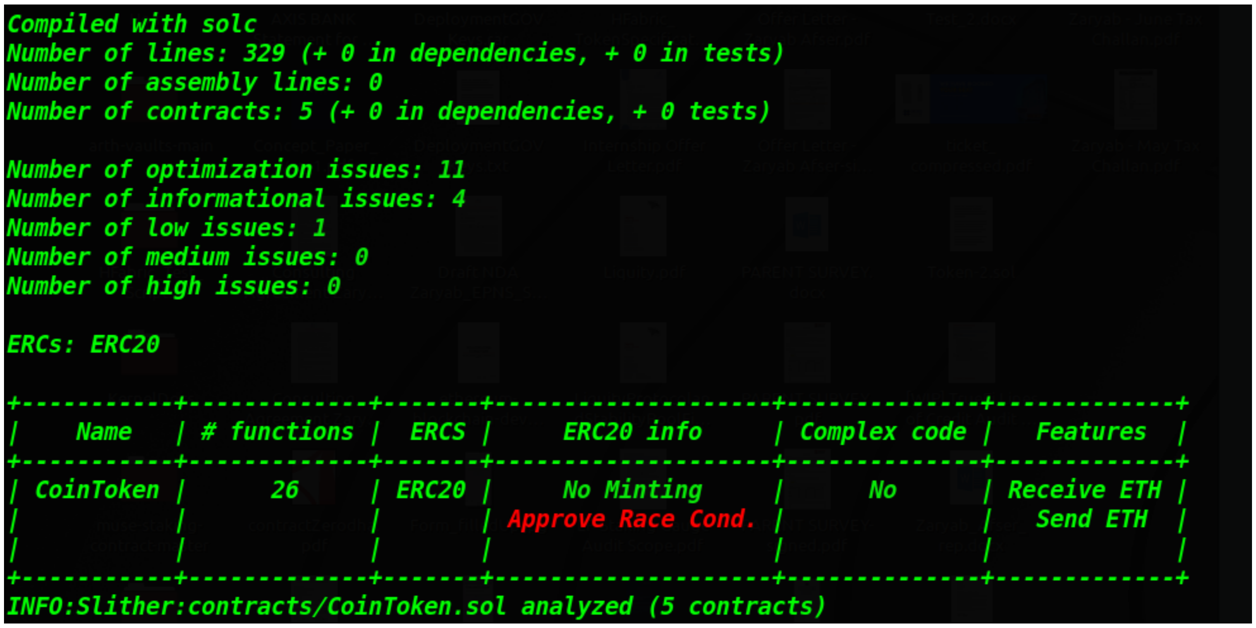
Informational
A.2 NatSpec Annotations must be included
Description:

The smart contracts do not include the NatSpec annotations adequately.

Recommendation:

It is considered a better practice to cover your smart contracts with NatSpec annotations.

Automated Test Results



```
Compiled with solc
Number of lines: 329 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 5 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 4
Number of low issues: 1
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC20

+-----------+-------------+-------+--------------------+--------------+-------------+
|    Name   | # functions |  ERCS |     ERC20 info     | Complex code |   Features  |
+-----------+-------------+-------+--------------------+--------------+-------------+
| CoinToken |      26     | ERC20 |     No Minting     |      No      | Receive ETH |
|           |             |       | Approve Race Cond. |              |   Send ETH  |
|           |             |       |                    |              |             |
+-----------+-------------+-------+--------------------+--------------+-------------+
INFO:Slither:contracts/CoinToken.sol analyzed (5 contracts)
```