Smart Contract Audit Report

for

R223Token

TRUSTLOOK

Version 1.0

Email: bd@trustlook.com

LAB BLOCKCHAIN

Project Overview

Project Name	R223Token
Contract codebase	N/A
Platform	Ethereum
Language	Solidity
Submission Time	2021.08.27

```
Symbol:                 R223
Name:                   R223Token

Circulating supply:     7 000 000 000
Total supply:           7 000 000 000
Max supply:             7 000 000 000
```

Report Overview

Report ID	TBL_20210724_00
Version	1.0
Reviewer	Blockchain Labs
Starting Time	2021.08.27
Finished Time	2021.08.28

Disclaimer

Audit reports do not provide any warranties or guarantees on the vulnerability-free nature of the given smart contracts, nor do they provide any indication of legal compliance. Audit process is aiming to reduce the high level risks possibly implemented in the smart contracts before the issuance of audit reports.

Audit reports can be used to improve the code quality of smart contracts and are not able to detect any security issues of smart contracts that will occur in the future.

Introduction

By reviewing the implementation of **R223Token** smart contracts, this audit report has been prepared to discover potential issues and vulnerabilities of their source code. We outline in the report about our approach to evaluate the potential security risks. Advice to further improve the quality of security or performance is also given in the report.

About R223Token

R223Token is an ERC-20 type smart contract intended for the creation and delivery of 7,000,000,000,000 R223 digital assets.

The address of the contract is:
0x428be91cb9a9093dfec1db9e85a6a04ac11dae5e

Upon creation, the contract immediately delivers the tokens to the creation wallet (R223: Deployer) with address:
0xe0D1EE8081EA438A57e9E1E1Eb0E5B2E7c2eDE2959 according to its Constructor:

```
// ----------------------------------------------------------------------
  // Constructor
  // ----------------------------------------------------------------------
  constructor() public {
    symbol = "R223";
    name = "R223Token";
    decimals = 6;
    _totalSupply = 7000000000000000;
    balances[0xe0D1EE8081EA438A57e9E1Eb0E5B2E7c2eDE2959] = _totalSupply;
    emit Transfer(address(0), 0xe0D1EE8081EA438A57e9E1Eb0E5B2E7c2eDE2959, _totalSupply);
  }
```

About Methodology

To evaluate the potential vulnerabilities or issues, we go through a checklist of well-known smart contracts related security issues using automatic verification tools and manual review. To discover potential logic weaknesses or project specific implementations, we thoroughly discussed with the team to understand the business model and reduce the risk of unknown vulnerabilities. For any discovered issue, we might test it on our private network to reproduce the issue to prove our findings.

The checklist of items is shown in the following table:

Category	Type ID	Name	Description
Coding Specification	CS-01	ERC standards	The contract is using ERC standards.
	CS-02	Compiler Version	The compiler version should be specified.
	CS-03	Construct or Mismatch	The constructor syntax is changed with Solidity versions. Need extra attention to make the constructor function right.
	CS-04	Return standard	Following the ERC20 specification, the transfer and approve functions should return a bool value, and a return value code needs to be added.
	CS-05	Address(0) validation	It is recommended to add the verification of require(_to!=address(0)) to effectively avoid unnecessary loss caused by user misuse or unknown errors.
	CV-06	Unused Variable	Unused variables should be removed.
	CS-07	Untrusted Libraries	The contract should avoid using untrusted libraries, or the libraries need to be thoroughly audited too.

	CS-08	Event Standard	Define and use Event appropriately
	CS-09	Safe Transfer	Using transfer to send funds instead of send.
	CS-10	Gas consumption	Optimize the code for better gas consumption.
	CS-11	Deprecated uses	Avoid using deprecated functions.
	CS-12	Sanity Checks	Sanity checks when setting key parameters in the system
Coding Security	SE-01	Integer overflows	Integer overflow or underflow issues.
	SE-02	Reentrancy	Avoid using calls to trade in smart contracts to avoid reentrancy vulnerability.
	SE-03	Transactio n Ordering Dependence	Avoid transaction ordering dependence vulnerability.
	SE-04	Tx.origin usage	Avoid using tx.origin for authentication.
	SE-05	Fake recharge	The judgment of the balance and the transfer amount needs to use the “require function”.
	SE-06	Replay	If the contract involves the demands for entrusted management, attention should be paid to the nonreusability of verification to avoid replay attacks.
	SE-07	External call checks	For external contracts, pull instead of push is preferred.
	SE-08	Weak random	The method of generating random numbers on smart contracts requires more considerations.
Additional Security	AS-01	Access control	Well defined access control for functions.
	AS-02	Authenticat ion management	The authentication management is well defined.
	AS-03	Semantic Consistenc y	Semantics are consistent.

The severity level of the issues are described in the following table:

Severity	**Description**
Critical	The issue will result in asset loss or data manipulations.
High	The issue will seriously affect the correctness of the business model.
Medium	The issue is still important to fix but not practical to exploit.
Low	The issue is mostly related to outedate, unused code snippets.
Informational	This issue is mostly related to code style, informational statements and is not mandatory to be fixed.

Audit Results

Here are the audit results of the smart contracts.

Scope

Following files have been scanned by our internal audit tool and manually reviewed and tested by our team:

File names	Compiler Version
R223Token.sol	v0.6.6+commit.6c089d02

This audit report is focused on the new update part of the new release.

Summary Details

- ID: TBL_SCA-001
- Severity: Informational
- Type: CS-10 (Gas consumption)
- Description:

The second validation of “amount > 0” is already done at 81. So this validation can be removed to save gas consumption.

- Remediation:

The dev team has updated the contract in the updated version with SHA1 value “2a2ec36ced889e44c11ca25bf2537dc1a6d92632”

- ID: TBL_SCA-002
- Severity: Informational
- Type: CS-12 (Sanity Checks)
- Description:

For key parameters in the system, it is recommended to add some sanity checks on update.

It is recommended to validate the parameter to be a non-zero value before the assignment.

- Remediation:

The dev team has updated the contract in the updated version with SHA1 value “2a2ec36ced889e44c11ca25bf2537dc1a6d92632”

- ID: TBL_SCA-003
- Severity: Informational
- Type: CS-08 (Event Standard)
- Description:

When defining an Event with address parameters, it is recommended to add “indexed” keyword for them for better query operations.

We advise to update these Events as follows:

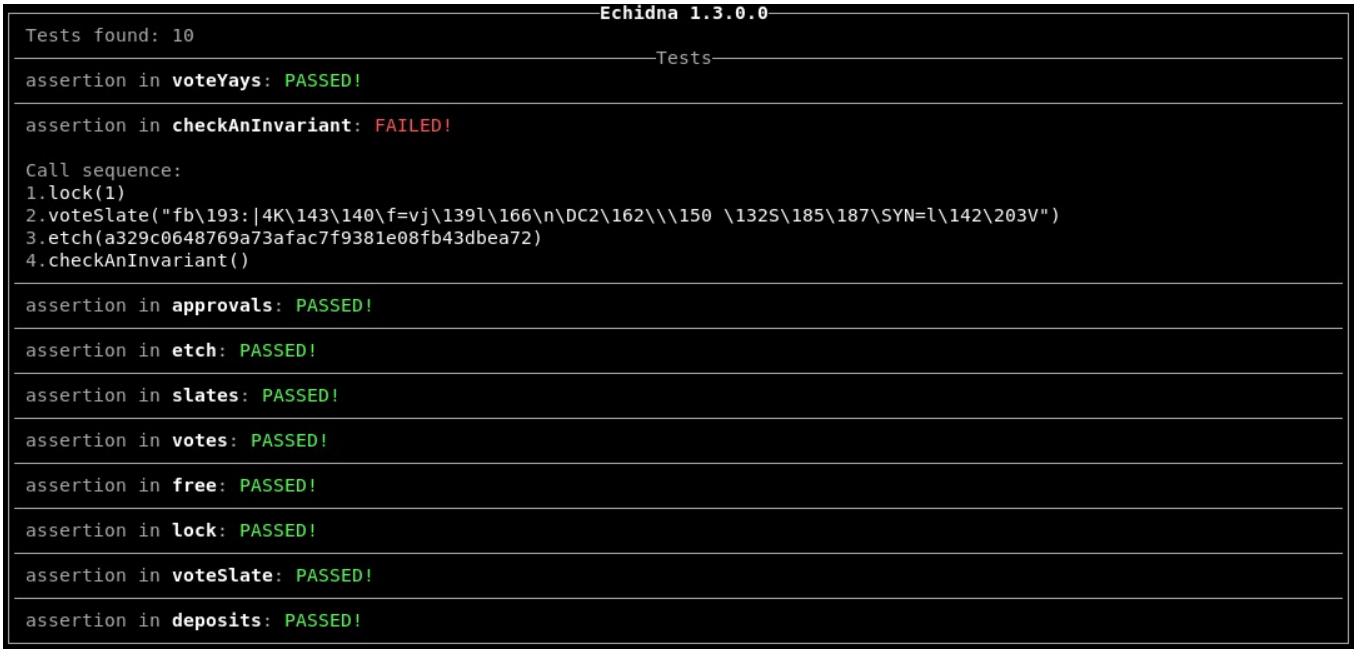
event Deposit(address indexed sender, address indexed asset, uint amount);
event Withdraw(address indexed sender, address indexed asset, uint amount);
event ReduceUnlockedAmount(address indexed depositor, address indexed asset, uint unlockedAmount);

- Remediation:

The dev team has updated the contract in the updated version with SHA1 value “2a2ec36ced889e44c11ca25bf2537dc1a6d92632

Auditor Comments

The audited smart contract can be deployed. Only low severity issues were found during the audit.



Conclusion

The audited smart contract can be deployed. Only low severity issues were found during the audit.

REVIEWED BY:

Blockchain Labs