Security Audit Report

DAOSquare Governance Token (RICE)

SECBIT

May 11, 2021

1. Introduction

DAOSquare Governance Token (RICE Token) is a token contract deployed on Ethereum. SECBIT Labs conducted an audit on May 11th, 2021, including an analysis of the contract in 3 areas: **code bugs**, **logic flaws**, and **risk assessment**. The assessment shows that the RICE Token contract has no critical security risks. The SECBIT team has some tips on logical implementation, potential risks, and code revising(see part 4 for details).

Type	Description	Level	Status
Code Revising	4.3.1 Several functions are internal and never called in the contract.	Info	No action required

2. Contract Information

This part describes the basic contract information and code structure.

2.1 Basic Information

The following list shows the basic information of RICE Token:

Name	**DAOSquare Governance Token**
Symbol	RICE
Decimals	18
Address	0xbD9908b0Cdd50386F92efCC8e1d71766C2782Df0
Lines	121
Source	Etherscan
Stage	Deployed

2.2 Contract List

The following content shows the contracts included in the RICE Token project:

Name	Lines	Description
IERC20	11	ERC20 interface
Context	10	Contract for accessing msg.sender and msg.data
ERC20	88	Implementation of ERC20 Token
RICEToken	6	Implementation of RICE Token

3. Contract Analysis

This part describes details of contract code assessment, including three items: the sum of tokens, authorities of contract accounts, functions of the contract.

3.1 TotalSupply

The sum of tokens in the contract is initialized to 100,000,000 by the constructor. The total supply is immutable after initialization.

3.2 Contract Account

There are two types of accounts in RICE Token: common account and approved account.

- Common Account
    - Description

      All accounts holding RICE Token
    - Authority
        - Transfer tokens in its own balance
        - Authorize other accounts to transfer its own token balance
    - Method of Authorization

      None
- Approved Account
    - Description

      Accounts authorized to transfer tokens from other accounts.
    - Authority
        - Transfer tokens from approved accounts (not exceeding approved sum)

- Method of Authorization

  Approved by other accounts

3.3 Feature Analysis

As a token contract, RICE Token meets with ERC20 contract standards. We can divide the key contract features into several parts:

- transfer / transferFrom

  Transferring is fundamental in a Token contract. RICE Token satisfies ERC20 and checks for security issues adequately. In solidity 0.8.0 and later on, overflows will be checked by default.

- approve

  An account can call `approve()` to allow other accounts managing its balance. ERC20 contract implements `increaseAllowance()` and `decreaseAllowance()` in addition, which enables avoiding re-approval attack.

4. Audit Detail

This part describes the process, and the detailed results of the audit also demonstrate the problems and potential risks.

4.1 Audit Process

The audit strictly followed the audit specification of SECBIT Lab. We analyzed the project from code bug, logical implementation, and potential risks. The process consists of four steps:

- Fully analysis of contract code line by line.
- Evaluation of vulnerabilities and potential risks revealed in the contract code.
- Communication on assessment and confirmation.
- Audit report writing.

4.2 Audit Result

After scanning with adelaide, sf-checker, and badmsg.sender (internal version) developed by SECBIT Labs and open source tools including Mythril, Slither, SmartCheck, and Securify, the auditing team performed a manual assessment. The team inspected the contract line by line, and the result could be categorized into twenty-one types:

Number	Classification	Result
1	Normal functioning of features defined by the contract	✓
2	No obvious bug (e.g., overflow, underflow)	✓
3	Pass Solidity compiler check with no potential error	✓

4	Pass common tools check with no obvious vulnerability	✓
5	No obvious gas-consuming operation	✓
6	Meet with ERC20	✓
7	No risk in low-level call (call, delegatecall, callcode) and in-line assembly	✓
8	No deprecated or outdated usage	✓
9	Explicit implementation, visibility, variable type, and Solidity version number	✓
10	No redundant code	!
11	No potential risk manipulated by timestamp and network environment	✓
12	Explicit business logic	✓
13	Implementation consistent with annotation and other info	✓
14	No hidden code about any logic that is not mentioned in design	✓
15	No ambiguous logic	✓
16	No risk threatening the developing team	✓
17	No risk threatening exchanges, wallets, and DApps	✓
18	No risk threatening token holders	✓
19	No privilege on managing others' balances	✓
20	No minting method	✓

21	Correct managing hierarchy	✓

4.3 Issues

4.3.1 Several functions are internal and never called in the contract.

Risk Type	Risk Level	Impact	Status
Code Revising	Info	Redundant code	No action required

Description

The contract implements basic functions of ERC20 Tokens, while some functions in the contract are internal and never called in public or external functions. For example, The internal function `_burn()` will never be called after the contract is deployed. Therefore, these functions are redundant codes, which may consume unnecessary gas when being deployed.

The RICE token adopts the standard ERC20 implementation from the OpenZeppelin team, which is a widespread practice.

5. Conclusion

After auditing and analyzing the RICE Token contract, SECBIT Labs found some issues to optimize and proposed corresponding suggestions, which have been shown above. SECBIT Labs holds the view that the RICE Token smart contract meets the standard of ERC20 and has high code quality.

Disclaimer

SECBIT smart contract audit service assesses the contract's correctness, security, and performability in code quality, logic design, and potential risks. The report is provided "as is", without any warranties about the code practicability, business model, management system's applicability, and anything related to the contract adaptation. This audit report is not to be taken as an endorsement of the platform, team, company, or investment.

APPENDIX

Vulnerability/Risk Level Classification

Level	Description
High	Severely damage the contract's integrity and allow attackers to steal ethers and tokens, or lock ethers inside the contract.
Medium	Damage contract's security under given conditions and cause impairment of benefit for stakeholders.
Low	Cause no actual impairment to contract.
Info	Relevant to practice or rationality of the smart contract, could possibly bring risks.

SECBIT Lab is devoted to construct a common-consensus, reliable and ordered blockchain economic entity.

http://www.secbit.io

audit@secbit.io

@secbit_io