QuillAudits

Audit Report
August, 2022

For

rova

FUTURE DIGITAL CASH

Table of Content

Executive Summary

Project Name	Rova
Overview	The RovaToken contract is a token creation contract. At the point of deployment, the initial total supply of the token is minted into the owner. The contract inherits the following contracts from the Openzeppelin standard; ERC20, Ownable, and ERc20Burnable. All of these aid in the minting and burning of tokens and also for ownership management.
Timeline	25 July, 2022 - 1 August, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Rova codebase for quality, security, and correctness. *https://github.com/ROVAToken/ROVA/blob/main/ROVA.sol* Commit hash: f344405beac7ded1611706d364f0d451d8970a53
Fixed in	*https://github.com/ROVAToken/ROVA/blob/main/ROVA.sol* Commit hash: e1928af061463886e3b45298fff1872916f3f93a

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	**2**	**2**

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility leve

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - Rova.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 Token Decimals

Description

The decimal of the token is 6 decimals. This becomes a problem when contracts that interact with Rova Token Contract, assume it is a 18 decimal token. In such a situation, there are possibilities of miscalculation that will yield unwanted outcomes. Here 1 token would be 1*(10**6) = 1000000 Wei. It may happen that any other smart contract uses/accepts this token for some reason. That smart contract calculates the token amount sent by the user assuming its 18 decimal token, which can result in unwanted outcomes.

Eg: Care needs to be taken in this type of scenario.

- User sends 1 token ("1*(10**6)" in this case) to a smart contract.
- The smart contract which accepts this token checks the token amount sent by User which was 1*(10**8) = 100000000
- While calculating amount sent by the User, smart contract uses 18 decimals and expects 1 token sent to be 1*(10**18) = 1000000000000000000

In this case this condition will fail since token amount sent by User is 1000000 i.e 1*(10**6) and not 1*(10**18)

Status

Resolved

A.2 Ownership Transfer must be a Two-step Processes

Description

Contracts are integrated with the standard Openzeppelin ownable contract, however, when the owner mistakenly transfers ownership to an incorrect address, ownership is completely removed from the original owner and cannot be reverted. The transferOwnership() function in the ownable contract allows the current owner to transfer his privileges to another address. However, inside transferOwnership() , the newOwner is directly stored in the storage, owner, after validating the newOwner is a non-zero address, which may not be enough.

Remediation

It would be much safer if the transition is managed by implementing a two-step approach: _transferOwnership() and _updateOwnership() . Specifically, the _transferOwnership () function keeps the new address in the storage, _newOwner , instead of modifying the _owner() directly. The updateOwnership() function checks whether _newOwner is msg.sender, which means _newOwner signs the transaction and verifies himself as the new owner. After that, _newOwner could be set into _owner.

Status

Resolved

Informational Issues

A.3 Unlocked pragma (pragma solidity ^0.8.14)

Description

Contract has a floating solidity pragma version. This is present also in inherited contracts. Locking the pragma helps to ensure that the contract does not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. The recent solidity pragma version also possesses its own unique bugs.

Remediation

Making the contract use a stable solidity pragma version prevents bugs occurrence that could be ushered in by prospective versions. It is recommended, therefore, to use a fixed solidity pragma version while deploying to avoid deployment with versions that could expose the contract to attack.

Status

Resolved

A.4 Remove Unused Imported Libraries

Description

The contract inherited a couple of libraries and contracts intended to aid achieve the safety creation of ERC20 token. However, there are some libraries that were left unused in the contract.

Remediation

It is recommended to remove unused libraries that are no longer needed to build the contract.

Status

Resolved

Functional Testing

Some of the tests performed are mentioned below

- Should get the name of the token
- Should get the symbol of the token
- Should get the symbol of the token
- Should get the total supply of the token when deployed
- Should get balance of the owner when contract is deployed
- Should transfer tokens to other address
- Should approve another account to spend token
- Should burn the token by an holder
- Should revert when trying to burn beyond balance
- Should mint to others address and increase total supply
- Should revert when non-owner calls the mint function
- Should Transfer and Update the ownership through current Owner

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Rova. We performed our audit according to the procedure described above.

Some issues of Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At The end. Rova Token Team Resolved all Issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Rova Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Rova Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
August, 2022

For

rova

FUTURE DIGITAL CASH

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com