QuillAudits

Audit Report
August, 2022

For

Table of Content

Executive Summary

Project Name	Ginoa
Overview	Ginoa Token is a simple ERC20 token contract that allows for the purchase of the Ginoa token with BNB; an Initial Coin Offering (ICO). It adopts the standard ERC20 standard. Ginoa contract has a stipulated time for the sales of the token.
Timeline	3 June, 2022 to 9 June, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze Ginoa codebase for quality, security, and correctness. *https://bscscan.com/token/0x2ff90b0c29ededdaf11c847925ea4a17789e88c3?a=0xa30d9098b4823e683eca120dcf24ae5ba011f893#readContract*

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	2	0	5
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	2	0

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
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

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

High Severity Issues

No issues found

Medium Severity Issues

1. Admin can withdraw during ICO

Line	Function - WithdrawToken

	`function WithdrawToken() public onlyOwner{`
	`  require(this.balanceOf(address(this)) > 0, "No Token Left");`
	`  this.transfer(msg.sender, this.balanceOf(address(this)));`
	`}`

Description

This function permits a malicious admin to make withdrawal of tokens during ICO.

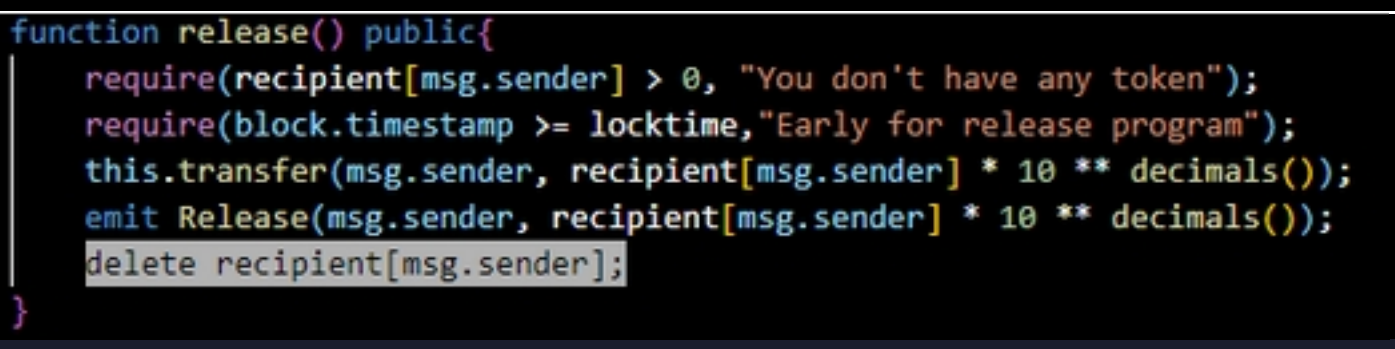
Remediation

In order to give investors a time window to act accordingly, it is advised that there should be a time lock feature to help moderate the power of the admin within the specific period of the ICO. A required check will help stipulate the time for withdrawal that won't deter the investors.

Status

Acknowledged

2. Time Period

Line	Function - release()
	

```
function release() public{
    require(recipient[msg.sender] > 0, "You don't have any token");
    require(block.timestamp >= locktime,"Early for release program");
    this.transfer(msg.sender, recipient[msg.sender] * 10 ** decimals());
    emit Release(msg.sender, recipient[msg.sender] * 10 ** decimals());
    delete recipient[msg.sender];
}
```

Description

The newTime parameter in the setTime() function lacks sufficient sanity value checks. A malicious admin can set it to a very long time, denying the recipients their tokens for a long time(resulting in a potential DOS attack). A non-unix timestamp can be accidentally set for lockTime via the setTime() function. An incorrect value setting to the newTime parameter can lead to all tokens getting instantly unlocked before the intended locktime.

Recommendation

Function should be designed to prevent passing in the wrong timestamp value which could prevent when a recipient can withdraw.

Status

Acknowledged

Low Severity Issues

3. Ownership Renouncement

Description

When the renounceOwnership function is called accidentally by the admin of the contract, the contract immediately renounces ownership to address zero, after which it makes it impossible for the owner to call the admin onlyOwner functions.

Remediation

It is recommended to override the renounce owner functionality to prevent the owner from calling this function or could fashion the renounce owner to work if one has successfully transferred ownership to the right address. If a multi-sig is to be utilized, it should be confirmed that two or more persons are required to sign before the execution of the renounce ownership function.

Status

Resolved

Auditor's Comment: Ginoa Team has renounced the ownership. *View Here.*

4. Transfer Ownership

Description

The transferOwnership() function in contract allows the current admin to transfer his privileges to another address. However, inside transferOwnership() , the newOwner is directly stored into the storage owner, after validating that the newOwner is a non-zero address, and immediately overwrites the current owner. This can lead to cases where the admin has transferred ownership to an incorrect address and wants to revoke the transfer of ownership or in the cases where the current admin get to know that the new admin has lost access to his account.

Remediation

Consider making transferOwnership a two step process in which the first step is to propose to the new owner the ownership of the contract and the second step is the ownership acceptance step in which the owner has to accept the ownership proposed to him.

Status

Resolved

Auditor's Comment: Ginoa Team has renounced the ownership. *View Here*.

Informational Issues

5. Unlocked pragma (pragma solidity ^0.8.9)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

The base contract has a floating solidity version. Contracts with an exact solidity version protect the contract from possible attacks that may arise due to new fixes and features that come with newer solidity compiler versions.

Status

Acknowledged

6. Absence of Comments (Natspec Format)

Description

There is an absence of comments in the contract. This comment, if it follows the Natspec format, will define every function and aid people interacting with the contract to understand what every line of code does. This gives a form of assurance and confidence on the project when users can detect the motive of a function from barely reading appropriate comments.

Remediation

Codes with appropriate Natspec comments are highly trusted and help others better understand the codes for smooth contract interaction.

Status

Acknowledged

7. Increment and Decrement of Phase Lock and Unlock

Description

When the release function is called by a recipient, the PhaseLocked is not being decreased and PhaseUnlocked is not being increased as recipients receive their tokens.

Remediation

This could be designed to increase and decrease the variable as recipients interact with the release function.

Status

Acknowledged

8. Variable Naming

Description

Price as a variable name is misleading. On increasing the price, the amount of tokens that can be bought increases(which is counter-intuitive to the notion that when price increases, less tokens can be bought) and vice versa.

Remediation

It is recommended to rename the variable accordingly to match the behavior of the variable in the code.

Status

Acknowledged

8. General Recommendation

This contract creates an ERC20 token with an integration of the initial coin offering (ICO). It allows for the sale of tokens for a particular period of time and gives the deployer of the contract the right to transfer tokens at a particular period of time and after the expiration of the locktime. It recommends that the TransferWithLock function be revisited to ensure that it carries out its principal action.

Status

Acknowledged

Functional Testing

Some of the tests performed are mentioned below

- Should get the name of the contract.
- Should get the symbol of the contract
- Should get owner of the contract
- Should get decimals of the contract
- Should get the locktime
- Should get total supply; the addition of ico and personal mint.
- Should get the owner's token balance
- Should show the zero balance of non-holder.
- Should estimate the token in the contract for ICO
- Should increase phase unlocked.
- Should own some token after buying.
- Should successfully withdraw by owner.
- Should revert when no owner calls the WithdrawToken function.
- Should revert when no owner calls the withdraw function.
- Should set price by the owner.
- Should revert when the price is set by the unknown.
- Should set time by the owner.
- Should revert when time is set by the unknown.
- Should get the value of an address after TransferWithLock
- Should increase the phase locked after TransferWithLock
- Should revert on calling TransferWithoutLock because amountToSend exceed address(this).
- Should increase the phase unlocked after TransferWithoutLock.
- Should revert if the caller has no token.
- Should revert if the function is called before locktime.

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 Ginoa Smart Contract. We performed our audit according to the procedure described above.

Some issues of Medium,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 Ginoa Team Fixed two Issues by renouncing Ownership and Acknowledged other issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Ginoa 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 Ginoa 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

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com