QuillAudits

Audit Report
August, 2021

CITRUS

Contents

Scope of Audit

The scope of this audit was to analyze and document the Citrus Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- 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
- BEP20 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 level

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 BEP-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 analyzed 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.
SmartCheck.

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 security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of 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 possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

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 to be fixed before moving to a live environment.

Medium level severity issues

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 level severity issues

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 four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	1	2
Acknowledged	0	0	1	2
Closed	0	0	0	0

Introduction

During the period of **August 11, 2021 to August 15, 2021** - QuillAudits Team performed a security audit for Citrus smart contracts.

The code for the audit was taken from the following official link: https://github.com/CitrusTech/CitrusTechContract/blob/master/CitrusToken.sol

Note	Date	Commit hash
Version 1	August	e50a2a983928c10b76e6bc374ae6267f51af99b6

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

Low level severity issues

1. Unnecessary use of require statement

Line	Code
111	require(amount > 0, "TimeLock: Amount cannot be zero");

Description

As the function can be only called by ‘Owner’, the use of require statement will only consume more gas, as an owner can instruct to not use 0 as a value while locking to waste Gas.

Remediation

Remove require statement to save GAS.

Status: Acknowledged by the Auditee.
Auditee Comments: The gas difference is negligible.

2. Does not use the value returned by external calls

Line	Code
127	BEP(address(this)).transfer(msg.sender, amount);

Description

The return value is not used when a function is returning a value.

Remediation
Use Require statement.

require(BEP(address(this)).transfer(msg.sender, amount));

Status: Open

Informational

3. Does not use safemath for operations

Description
Safemath operation is missing where operators are used directly without considering overflow and underflow.

Remediation
Use Safemath at all operations.

Status: Acknowledged by the Auditee.
Auditee Comments: We don't have any such functionality where Safemath is required, so we didn't use it.

4. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:

- wned.changeOwnership (./citrus.sol#10-12) should be declared external
- BEP20.balanceOf (./citrus.sol#38) should be declared external
- BEP20.transferFrom (./citrus.sol#48-55) should be declared external
- BEP20.approve (./citrus.sol#57-61) should be declared external
- BEP20.allowance (./citrus.sol#63-65) should be declared external
- TimeLock.timelock (./citrus.sol#110-118) should be declared external
- TimeLock.release (./citrus.sol#120-133) should be declared external
- TimeLock.lockedAccountDetails (./citrus.sol#135-151) should be declared external

Remediation
Use the external attribute for functions that are never called from the contract.

Status: Open

5. Use Transfer Event in Constructor

Description
It will be safer if the mint function is used in the Constructor, to mint the initial supply of tokens instead of directly updating the balances and total supply variables.

Remediation
emit Transfer(address(0), account, amount);

Status: Open

6. Use Require statement for multiple checks in transfer event

Description
The transfer function in the BEP20 contract is missing the require statements.

Remediation
a) In function transfer(address _to, uint256 _amount) , missing:
require(_to != address(0), "BEP20: transfer from the zero address");

b) In function transferFrom(address _from,address _to,uint256 _amount)
require(_from != address(0), "BEP20: transfer from the zero address");
require(_to != address(0), "BEP20: transfer to the zero address");
which is advisable to add in the beginning of both the fuctions.

Status: Acknowledged by the Auditee.
Auditee Comments: We have taken care of ‘require’ at the consumer level.

Functional test

Function Names	Technical Result	Logical Result	Overall Result
Read Functions()			
allowance	Passed	Passed	Passed
balanceOf	Passed	Passed	Passed
decimal	Passed	Passed	Passed
lockedAccountDetails	Passed	Passed	Passed
name	Passed	Passed	Passed
owner	Passed	Passed	Passed
symbol	Passed	Passed	Passed
totalSupply	Passed	Passed	Passed
Write Functions()			
approve	Passed	Passed	Passed
burn	Passed	Passed	Passed
changeOwnership	Passed	Passed	Passed
mint	Passed	Passed	Passed
transfer	Passed	Passed	Passed
transferFrom	Passed	Passed	Passed

Automated Testing

Slither

```
TimeLock.release (./citrus.sol#120-133) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= (loc.time + loc.lockedAt),TimeLock: Release time not reached) (./citrus.sol#125)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#block-timestamp
INFO:Detectors:
Owned.changeOwnership (./citrus.sol#10-12) should be declared external
BEP20.balanceOf (./citrus.sol#38) should be declared external
BEP20.transferFrom (./citrus.sol#48-55) should be declared external
BEP20.approve (./citrus.sol#57-61) should be declared external
BEP20.allowance (./citrus.sol#63-65) should be declared external
TimeLock.timelock (./citrus.sol#110-118) should be declared external
TimeLock.release (./citrus.sol#120-133) should be declared external
TimeLock.lockedAccountDetails (./citrus.sol#135-151) should be declared external
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#public-function-that-could-be-declared-as-external
INFO:Detectors:
Detected issues with version pragma in ./citrus.sol:
        - pragma solidity0.5.16 (./citrus.sol#1): it allows old versions
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#incorrect-version-of-solidity
INFO:Detectors:
Parameter '_newOwner' of Owned.changeOwnership (./citrus.sol#10) is not in mixedCase
Parameter '_owner' of BEP20.balanceOf (./citrus.sol#38) is not in mixedCase
Parameter '_to' of BEP20.transfer (./citrus.sol#40) is not in mixedCase
Parameter '_amount' of BEP20.transfer (./citrus.sol#40) is not in mixedCase
Parameter '_from' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_to' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_amount' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_spender' of BEP20.approve (./citrus.sol#57) is not in mixedCase
Parameter '_amount' of BEP20.approve (./citrus.sol#57) is not in mixedCase
Parameter '_owner' of BEP20.allowance (./citrus.sol#63) is not in mixedCase
Parameter '_spender' of BEP20.allowance (./citrus.sol#63) is not in mixedCase
Function 'BEP20._mint' (./citrus.sol#67-75) is not in mixedCase
Function 'BEP20._burn' (./citrus.sol#78-89) is not in mixedCase
Function 'BEP20._beforeTokenTransfer' (./citrus.sol#90) is not in mixedCase
Parameter '_lockAccount' of TimeLock.timelock (./citrus.sol#110) is not in mixedCase
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
TimeLock.release (./citrus.sol#120-133) does not use the value returned by external calls:
        -BEP(address(this)).transfer(msg.sender,amount) (./citrus.sol#127)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#unused-return
INFO:Detectors:
TimeLock.release (./citrus.sol#120-133) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= (loc.time + loc.lockedAt),TimeLock: Release time not reached) (./citrus.sol#125)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#block-timestamp
INFO:Detectors:
Owned.changeOwnership (./citrus.sol#10-12) should be declared external
BEP20.balanceOf (./citrus.sol#38) should be declared external
BEP20.transferFrom (./citrus.sol#48-55) should be declared external
BEP20.approve (./citrus.sol#57-61) should be declared external
BEP20.allowance (./citrus.sol#63-65) should be declared external
TimeLock.timelock (./citrus.sol#110-118) should be declared external
TimeLock.release (./citrus.sol#120-133) should be declared external
TimeLock.lockedAccountDetails (./citrus.sol#135-151) should be declared external
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#public-function-that-could-be-declared-as-external
INFO:Detectors:
Detected issues with version pragma in ./citrus.sol:
        - pragma solidity0.5.16 (./citrus.sol#1): it allows old versions
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#incorrect-version-of-solidity
INFO:Detectors:
Parameter '_newOwner' of Owned.changeOwnership (./citrus.sol#10) is not in mixedCase
Parameter '_owner' of BEP20.balanceOf (./citrus.sol#38) is not in mixedCase
Parameter '_to' of BEP20.transfer (./citrus.sol#40) is not in mixedCase
Parameter '_amount' of BEP20.transfer (./citrus.sol#40) is not in mixedCase
Parameter '_from' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_to' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_amount' of BEP20.transferFrom (./citrus.sol#48) is not in mixedCase
Parameter '_spender' of BEP20.approve (./citrus.sol#57) is not in mixedCase
Parameter '_amount' of BEP20.approve (./citrus.sol#57) is not in mixedCase
Parameter '_owner' of BEP20.allowance (./citrus.sol#63) is not in mixedCase
Parameter '_spender' of BEP20.allowance (./citrus.sol#63) is not in mixedCase
```

Results

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

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Citrus 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 Citrus Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

Some low severity issues were detected; it is recommended to fix them.

CITRUS

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com