QuillAudits

Audit Report
September, 2021

For

ALFCOIN

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Alfcoin 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
- 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 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 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
Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

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

Risk-level	Description
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 four 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.

Number of issues per severity

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

Introduction

During the period of **September 23, 2021, to September 25, 2021** - QuillAudits Team performed a security audit for **Alfcoin** smart contracts.

The code for the audit was taken from following the official link: https://bscscan.com/address/0xD185F089c3A7f013dC08fdD8BaE812909422f393#code

Issues Found

A. Contract – Alfcoin

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational Issues

1. **Unlocked pragma directives**

```
pragma solidity ^0.8.0;
```

Description
Contracts should be deployed using the same compiler version/flags with which they have been tested. Locking the pragma (e.g. by not using ^ in *pragma solidity* 0.8.0) ensures that contracts do not accidentally get deployed using an older compiler version with unfixed bugs.

Remediation
Lock the pragma version.

Status: Acknowledged by the Auditee

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
transfer()	Passed
allowance()	Passed
approve()	Passed
transferFrom()	Passed
increaseAllowance()	Passed
decreaseAllowance()	Passed

Automated Tests

Slither

```
Context._msgData() (bep-20.sol#23-25) is never used and should be removed
ERC20._burn(address,uint256) (bep-20.sol#418-433) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (bep-20.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (bep-20.sol#32) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (bep-20.sol#117) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (bep-20.sol#148) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (bep-20.sol#505) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract alfcoin (bep-20.sol#508-514) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

name() should be declared external:
        - ERC20.name() (bep-20.sol#205-207)
symbol() should be declared external:
        - ERC20.symbol() (bep-20.sol#213-215)
decimals() should be declared external:
        - ERC20.decimals() (bep-20.sol#230-232)
totalSupply() should be declared external:
        - ERC20.totalSupply() (bep-20.sol#237-239)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (bep-20.sol#244-246)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (bep-20.sol#256-259)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (bep-20.sol#264-266)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (bep-20.sol#275-278)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (bep-20.sol#293-307)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (bep-20.sol#321-324)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (bep-20.sol#340-348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
bep-20.sol analyzed (5 contracts with 75 detectors), 20 result(s) found
```

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.

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.

Disclaimer

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

Audit Report September, 2021

For

ALFCOIN

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com