QuillAudits

Audit Report
April, 2022

For

COINFANTASY

Table of Content

Executive Summary

Project Name	CoinFantasy
Overview	World's first decentralized fantasy trading game for crypto markets
Timeline	15 June, 2022 - 27 June, 2022
Method	Manual Review, Functional Testing, Automated Testing, etc.
Scope of Audit	The scope of this audit was to analyse **CoinFantasy (betacoinfantasy)** codebase for quality, security, and correctness.
Source Code	*https://github.com/Coinfantasyio/cf-smart-contracts/tree/dev*
Fixed In	*56484de05fb49b83c283ad3d6322176479edc91e*

High Medium

Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Closed Issues	1	0	1	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
- 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 - BetaCoinfantasyToken.sol

High Severity Issues

A.2 Any active and approved ERC223 recipient can burn tokens of other users

Line	Code

79-86	function burnFrom(address user, uint256 _amount) public virtual onlyERC223Recipient {
	balances[user] = balances[user].sub(_amount);
	_totalSupply = _totalSupply.sub(_amount);
	
	bytes memory empty = hex"00000000";
	emit OnERC223TokenBurned(user, user, _amount, balances[user], user);
	emit Transfer(user, address(0), _amount, empty);
	}

Description

The burnFrom function has a modifier which checks if the msg.sender is an active and approved ERC223 recipient. This means any active and approved ERC223 recipient can use this function and burn tokens of any other user.

Remediation

It's recommended that you only let an authorized entity (owner or minter) burnFrom other user accounts. Also check isApprovedERC223Recipient for the user whose tokens are to be burnt.

Status

Fixed

Medium Severity Issues

No issues were found

Low Severity Issues

A.2 ERC223 standard not followed properly

Description

The contracts for the token inherit from the ERC223 token contract, but it's missing some of the important functions like name() and symbol(). Not following the token standard might result in explorer websites not able to list the token properly. There can be issues for websites displaying the token metadata.

Remediation

It's recommended that you implement all the functions listed in the standard.
Please follow this for reference :
https://github.com/Dexaran/ERC223-token-standard/blob/development/token/ERC223/IERC223.sol

Status

Fixed

Informational Issues

No issues were found

Functional Test

Some of the tests performed are mentioned below

- Should not be able to transfer or transferFrom
- Minter should be able to mint
- Owner should be able to add new recipients
- Owner should be able to remove recipients
- Should be able to burn tokens if active and valid recipient

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

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,some issues of High and Low severity were found, which the coin fantasy team resolved.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **CoinFantasy 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 CoinFantasy 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
April, 2022

For

COINFANTASY

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com