QuillAudits

Audit Report
February, 2024

For

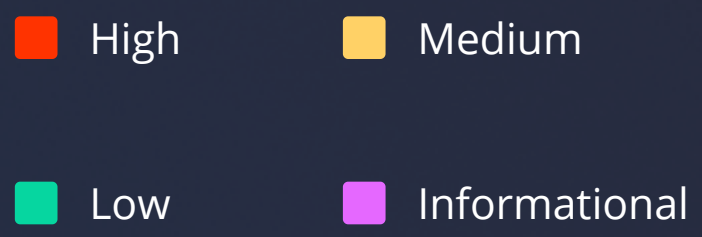
QUANTELICA

Table of Content

Executive Summary

Project Name	Quantelica
Project URL	*https://quantelica.com/*
Overview	Quantelica is a gaming ecosystem tailored to the needs of Web3 gaming. The platform is designed to integrate third-party games, provide cutting-edge software development kits (SDKs), and offer unique NFT capabilities. It will feature a marketplace, a community, unique NFT collections, play-to-earn games, and incentives for players. Audited Contracts Overview:- 1]Contract PioneerCollection: allows users to mint NFTs from different subcollections, each with its own unique properties and characteristics. Supports GSN and Chainlink VRF. 2]Contract PartnershipProcessor: Implements the logic for processing partnershipoperations, including discount application, commission distribution to various participants (partners, shareholders), and management of DApp project settings.
Audit Scope	*https://git.quantelica.com/blockchain/audit-271122*
Contracts in Scope	Branch: Main Contracts:- -PioneersCollection.sol -PartnershipProcessor.sol and interfaces
Commit Hash	f45b4346e82b9441997687d9147c008a898ef7ac
Language	Solidity
Blockchain	Ethereum
Method	Manual Review, Automated Tools, Functional Testing
Review 1	11th January 2024 - 22nd January 2024
Fixed In	NA

Number of Security Issues per Severity



	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	0	0

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw

Checked Vulnerabilities

- Using inline assembly
- Unsafe type inference
- Style guide violation
- Implicit visibility level

Techniques and Methods

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

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC 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.

Static Analysis

A 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 analyzed, 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 behavior 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, Solhint, Mythril, Slither, Solidity Static Analysis.

Types of Severity

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.

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 be fixed before moving to a live environment.

Medium 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 Severity Issues

Low-level severity issues can cause minor impact and 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.

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.

A. Smart Contracts Manual Analysis

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Informational Issues

No issues were found.

Functional Tests

Some of the tests performed are mentioned below:

PioneersCollection:

- ✓ Should allow the owner to update referral fees
- ✓ Should revert if a non-owner tries to update referral fees
- ✓ Should handle referral fees with discount correctly

Collection Modification Access:

- ✓ Should revert if an unauthorized address tries to add a sub-collection
- ✓ Should show the correct URI after the mint
- ✓ Should not mint without proof, when whitelist is enabled
- ✓ Should mint with pseudo-random ID assignment
- ✓ Should disable whitelist for minting and mint token without proof
- ✓ Should sell all tokens

Access Control Tests:

- ✓ Should allow the owner to add a sub-collection
- ✓ Should not allow the owner to add a duplicate sub-collection
- ✓ Should mint a token and assign it to the sender
- ✓ Should transfer a token from one address to another
- ✓ Should revert on available main supply exceeded
- ✓ Should revert on max supply pre address exceeded
- ✓ Should set the trusted forwarder
- ✓ Should set the mint price
- ✓ Should set the maximum number of tokens that can be minted by a single address
- ✓ Should set the whitelist minting option
- ✓ Should enable/disable the burn functionality
- ✓ Should set the royalty information

Functional Tests

- ✓ Should return the correct sub-collection ID by token ID
- ✓ Should revert when sending to the blacklisted address
- ✓ Should revert when approving to the blacklisted address
- ✓ Should revert when no nested collections

ReferralProcessor

- ✓ Should set project and register owner pools
- ✓ Should process referral operation with discount when eligible
- ✓ Should distribute funds according to owner pools
- ✓ Should remove owner pools

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 Quantelica. We performed our audit according to the procedure described above.

Quantelica Codebase is good to Deploy.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Quantelica smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Quantelica smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Quantelica to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

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.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
February, 2024

For

QUANTELICA

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com