QuillAudits

AUDIT REPORT

April , 2025

For

GRQ

GET RICH QUICK INSTITUTE

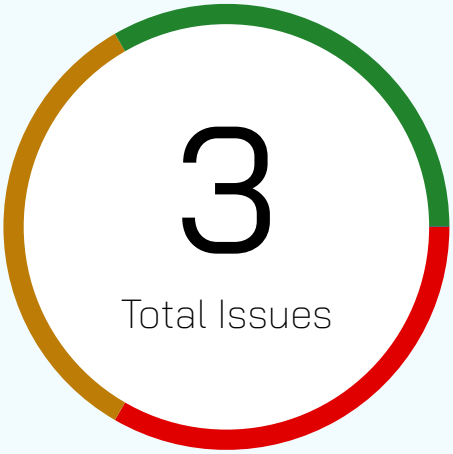
Table of Content

Executive Summary

Project name	GRQ
Project URL	https://docs.grq.institute/
Overview	GenesisRewardPool smart contract implements a short-term (7-day) staking mechanism to bootstrap the initial supply of a token. Users can deposit tokens into various pools to earn rewards, with each pool having configurable allocation points and deposit fees. The contract includes features for single or batch deposits/withdrawals, emergency withdrawals, and a claim-all function to collect rewards from multiple pools at once. The operator role has administrative privileges to add/modify pools and recover funds.
Audit Scope	The scope of this Audit was to analyze the GRQ Smart Contracts for quality, security, and correctness.
Source Code link	https://github.com/grq-institute/core-contracts/blob/master/src/farms/GenesisRewardPool.sol
Contracts in Scope	GenesisRewardPool.sol
Branch	main
Commit Hash	b18f168
Language	Solidity
Blockchain	EVM

Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	4th April 2025 - 6th April 2025
Updated Code Received	6th April 2025
Review 2	6th April 2025 - 7th April 2025
Fixed In	https://github.com/grq-institute/core-contracts/commit/afe23bde66b76edeeebf47599ca9975bef770506
Mainnet address	0x4E2eed2cFfAf9bCADF0c874E05a524AB69Be0a3B

Number of Issues per Severity



High	1 (33.33%)
Medium	1 (33.33%)
Low	1 (33.33%)
Informational	0 (0.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	**1**	**1**	**1**	0
Acknowledged	0	0	0	0
Partially Resolved	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 Computations 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

- [x] Using inline assembly
- [x] Unsafe type inference
- [x] Style guide violation
- [x] 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, Foundry, Solhint, Mythril, Slither, Solidity statistical 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 Issues

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

Security vulnerabilities identified that must be resolved and are currently unresolved.

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.

High Severity Issues

The contract increases the global rewardsPerSecond each time a pool starts, leading to more reward distributions.

Resolved

Path

GenesisRewardPool.sol

Function

add()

Description

The contract increases the global rewardsPerSecond each time a pool starts, leading to more reward distributions. Each new pool adds its allocation points to rewardsPerSecond, causing the total rewards to multiply with each pool addition.
This could allow protocol to distribute significantly more rewards than intended.

Recommendation

Define a fixed rewardsPerSecond and remove the reward rate modifications from updatePool and add functions.

Medium Severity Issues

The governanceRecoverUnsupported function lacks a reward token check.

Resolved

Path

GenesisRewardPool.sol

Function

governanceRecoverUnsupported()

Description

There is no check for the input token as rewardToken in the recover function. While protected by timelock and multisig, this still remains an issue that could allow reward tokens to be recovered, potentially affecting user rewards.

Recommendation

Add a check for the rewardToken in the function.

Low Severity Issues

No check against zero address in setOperator function

Resolved

Path

GenesisRewardPool.sol

Function

setOperator

Description

No check against zero address in setOperator function , setting operator to zero address would permanently disable the access to call the operator functions .

Recommendation

Add a check for zero address in the setOperator.

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

Issues of High,Medium, low severity was found. In the End,GRQ Team Resolved all Issues.

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

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

QuillAudits

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

April , 2025

For

GRQ
GET RICH QUICK
INSTITUTE

QuillAudits

Canada, India, Singapore, UAE, UK
www.quillaudits.com audits@quillaudits.com