QuillAudits

AUDIT REPORT

May , 2025

For

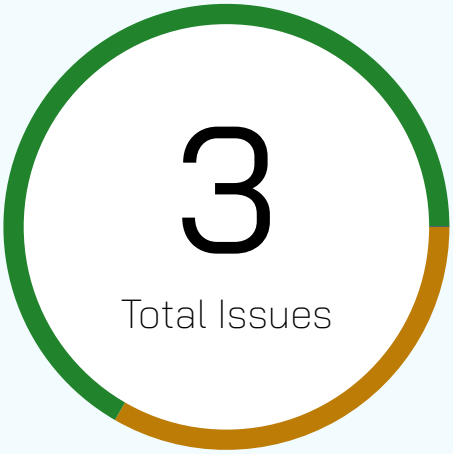
G.O.A.T.AI

Table of Content

Executive Summary

Project name	GOAT AI
Project URL	https://www.goatathletics.ai/
Overview	The GOATAITreasury contract manages token withdrawals and swaps via Uniswap V2, enabling controlled treasury management with role-based access and upgradeability.
Audit Scope	https://github.com/GOAT-Athletics-Inc/contracts/blob/feat/goatai-treasury/contracts/GOATAITreasury.sol
Contracts in Scope	GOATAITreasury.sol
Commit Hash	2415f11e8b51ceb15c13742c7fee1a996fdbcf7d
Language	Solidity
Blockchain	EVM
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	11 April 2025 - 21 April 2025
Updated Code Received	4 May 2025
Review 2	4 May 2025
Fixed In	554324fc011fe620c32cce9a579c8062695f8a48

Number of Issues per Severity



High	0 (0.00%)
Medium	1 (33.33%)
Low	2 (66.67%)
Informational	0 (0.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	**1**	**1**	0
Acknowledged	0	0	**1**	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.

Medium Severity Issues

Incorrect Swap Path Generation for WETH BaseTokens

Resolved

Path

GOATAITreasury.sol

Function

_getSwapParams()

Description

The _getSwapParams() function in the GOATAITreasury contract constructs a fixed three-token path for all swaps: [baseToken, WETH, outputToken]. While this is generally appropriate for routing through a common intermediary like WETH, it introduces a critical flaw when the baseToken is already WETH. In such cases, the generated path becomes [WETH, WETH, outputToken].
Uniswap V2 explicitly disallows creating or using liquidity pairs with identical token addresses, as enforced in its createPair() function. This can lead to failed transactions, revert errors, or unexpected swap behavior.

Recommendation

To maintain compatibility with Uniswap V2 and prevent unnecessary or invalid swap paths, the contract should conditionally construct the path.

Low Severity Issues

Hardcoded Swap Timeout Parameter Limits User Control and Creates Operational Vulnerabilities

Resolved

Path

GOATAITreasury.sol

Function

withdrawWithSwap()

Description

The contract's withdrawWithSwap function currently uses a hardcoded transaction deadline of block.timestamp + 300 (5 minutes) when calling Uniswap's swap function. This rigid approach limits flexibility and creates potential problems during network congestion,

where transactions might not be processed within this fixed timeframe, leading to failed swaps. Additionally, using a fixed deadline makes transactions more predictable and potentially vulnerable to MEV attacks.

Recommendation

A better approach would be adding a deadline parameter to the function

Centralization Risk

Acknowledged

Path

GOATAITreasury.sol

Function

-

Description

The GOATAITreasury contract grants significant authority to the DEFAULT_ADMIN_ROLE, including control over upgrades, role assignments, pausing, and key configuration changes. This creates a centralization risk where a single compromised admin account could take full control of the protocol, manipulate treasury funds, change critical parameters, or permanently pause functionality. The excessive permissions concentrated in a single role represent a significant security vulnerability as there are no checks and balances to prevent misuse of admin privileges.

Recommendation

Use governance mechanisms to make the contract more robust.

Functional Tests

Some of the tests performed are mentioned below:

- withdrawWithSwap fails for executor when treasury is paused
- withdrawWithSwap fails for charityWallet when treasury is paused
- Ensures withdrawal of other tokens fails for charityWallet when paused
- Attempting to withdraw more tokens than available in the treasury reverts with Insufficient-Balance error
- Random users without proper roles cannot withdraw tokens from the treasury
- Withdrawal reverts when provided slippage tolerance exceeds allowed limits
- Attempting to withdraw zero tokens reverts with InvalidWithdrawalAmount error
- Ensures withdrawDirect reverts for charityWallet during pause
- Ensures withdrawDirect reverts for executor during pause
- setting the Uniswap router is not allowed when treasury is paused
- Ensures grantRole fails during pause even when called by charityWallet
- charityWallet (admin) can unpause and re-pause the treasury
- Admin can withdraw other ERC20 tokens from the treasury

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 GOAT AI. We performed our audit according to the procedure described above.

Some issues of Medium and Low severity were found. Some suggestions, gas optimizations and best practices are also provided in order to improve the code quality and security posture.GOAT AI team resolved almost all and acknowledged one.

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

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

Follow Our Journey

AUDIT REPORT

May , 2025

For

G.O.A.T.AI

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com