QuillAudits

Audit Report
September, 2023

For

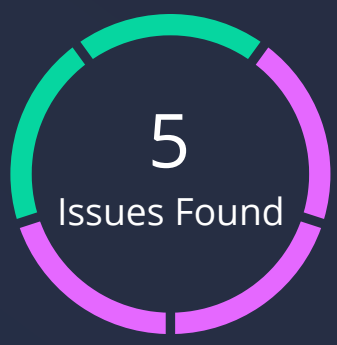
MOI

Table of Content

Executive Summary

Project Name	MOI
Project URL	*https://moi.technology/*
Overview	MoiIncentiveContract is an allocation management system designed to allocate a proportion of token assets to an allocation class. A single class has a primary identifier in form of bytes1, the name of the class, the maximum capacity of tokens that can be allotted to the class, a figure that tracks how much allocation has been made into the class, a metadata information that describes the the allocation class and then the status that notes the activeness or inactiveness of the allocation classes.
Audit Scope	*https://github.com/moinetworkllc/airdrop-eth-contract/blob/develop/contracts/MoiIncentiveContract.sol*
Contracts in Scope	MoiIncentiveContract
Commit Hash	bbcd43518fed5c0fd9fa2454216aac12f0f6882a
Language	Solidity
Blockchain	Ethereum
Method	Manual Analysis, Automated Testing, Functional Testing
Review 1	2 August 2023 - 9 August 2023
Updated Code Received	25 August 2023
Review 2	4 september 2023
Fixed In	91746fa744dcb52dde313a343b3925ce64a93746

The Number of Security Issues per Severity



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	1	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	1	3

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
- Style guide violation
- Unsafe type inference
- 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 behaviour.
- Token distribution and calculations are as per the intended behaviour 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 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

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. Contract - MoiIncentiveContract

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 Remove pragma experimental ABIEncoderV2

Line

3

Function

```
pragma solidity ^0.8.17;
pragma experimental ABIEncoderV2;
```

Description

In older solidity versions, there were associated issues of solidity not supporting arbitrary nested arrays as function arguments and also the return of dynamic nested arrays. This was one of the semantic and syntactic changes introduced in version 0.5.0. This feature, a research discovered, introduced some bugs in live production code. As a remedy and one of the breaking changes introduced in solidity version 0.8.0 and above, this has been rectified to be a standard that is activated by default. However, according to the solidity documentation, this could make some function calls more expensive and also facilitate some contract calls to revert that would ordinarily not revert in ABI coder v1 when they do not conform to some parameters type.

Remediation

With the usage of solidity version 0.8.17 for this contract, it is recommended to remove this from the contract since it is activated by default or if this is to be retained, use in accordance with the solidity doc, by using pragma abicoder v2;.

Status

Resolved

A.1 Remove pragma experimental ABIEncoderV2

References

- *https://soliditylang.org/blog/2019/03/26/solidity-optimizer-and-abiencoderv2-bug/*
- *https://docs.soliditylang.org/en/v0.8.0/080-breaking-changes.html*
- *https://stackoverflow.com/questions/72096399/warning-experimental-features-are-turned-on-do-not-use-experimental-features-o*
- *https://ethereum.stackexchange.com/questions/58698/typeerror-this-type-is-only-supported-in-the-new-experimental-abi-encoder*

Informational Issues

A.2: Lengthy Error Messages

Description

Error handlings are very much important to users to help them understand when a transaction reverts supposing it does not meet an expected requirement. However, when the require type exception is used with a lengthy error message, it facilitates the consumption of more gas.

Remediation

Use custom errors to minimize gas or use an abridged error message.

Status

Resolved

References

- *https://bitsbyblocks.com/solidity-tips-and-tricks-3-custom-errors/*
- *https://soliditylang.org/blog/2021/04/21/custom-errors/*

A.3: Unlocked pragma (pragma solidity ^0.8.17)

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Resolved

A.4 General Recommendation

In the MoiIncentiveContract, it uses the bytes1 data type as the primary identifier to differentiate between allocation classes. New identifiers for the new allocation class are computed from bytes1 newIndex = bytes1(uint8(allocationClassIndices.length)) in the createAllocationClass function. This is fair enough since the max of uint8 is 2^8 - 1 resulting in 255. This implies that the identifiers range from 0x00 - 0xff . Although the client affirmed that allocation classes will be in the 10s of number, it is important to note that if at any point the allocation class exceeds 255, this will cause an arithmetic overflow and newly created allocation classes to override previous classes.

Status

Resolved

Functional Tests

Some of the tests performed are mentioned below:

- Should allow contract owner to create allocation class with the class name, its capacity and metadata
- Should allow contract owner to update the capacity of the allocation class
- Should revert when the new capacity amount is not greater than the previous capacity amount
- Should allow the contract owner to update the status of the allocation class
- Should allow the contract owner to update the metadata of the allocation class
- Should allow the contract owner to set the allocation rate limit per allocation
- Should allow the contract owner to allocate tokens amount to users using their users MOI ID, amounts and allocation proof hash
- Should check for all possible reverts during allocation
- Should get allocation proof hash, users allocation, and all other getter functions

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

Some issues of low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in MOI 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 MOI 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 MOI 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

800K
Lines of Code Audited

Follow Our Journey

Audit Report
September, 2023

For

MOI

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com