

Moxie Findings & Analysis Report

Published: Sep 25, 2024

Prepared for: Moxie

Prepared by: Code4rena

Audit dates: Sep 23-Sep 24, 2024

Contents

- 1. Overview
 - 1.1 About C4
 - 1.2 About Moxie
- 2. Summary
- 3. Scope
- 4. Severity Criteria
- 5. Audit Timeline
- 6. Informational Findings
 - 6.1. `unlockTimeInSec_ ` calculation can be abstracted
 - 6.2. Add tests for duplicate indexes
 - 6.3. Redundancies
 - 6.4. Share buyer isn't correctly logged for `buyAndLockFor()` and `buyAndLockMultipleFor()`

1. Overview

1.1 About C4

Code4rena (C4) is an open organization consisting of security researchers, auditors, developers and individuals with domain expertise in smart contracts.

A C4 audit is an event in which community participants, referred to as Wardens, review, audit or analyze smart contract logic in exchange for a bounty provided by sponsoring projects

During the audit outlined in this document, C4 conducted an analysis of the Moxie smart contract system written in undefined. The audit took place from Sep 23 to Sep 24, 2024.

1.2 About Moxie

Moxie is an orchestration of several smart contracts that can be executed via Frames, Actions, and Apps/Clients. It represents foundational technology that anyone can use to add economic incentives to their Farcaster experience.

2. Summary

SEVERITY	COUNT
Critical	0
High	0
Medium	0
Low	0
Informational	4

3. Scope

The source code was delivered to Code4rena in a public Git repository.



4. Severity Criteria

C4 assesses the severity of disclosed vulnerabilities based on the primary risk categories: high, medium, low and informational.

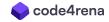
High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

For more information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4 website, specifically our section on Severity Categorization.

5. Audit Timeline

DATE	EVENT
Sep 23, 2024	Kick-off call
Sep 23, 2024	Audit start
Sep 24, 2024	Audit end



6. Informational Findings

A total of 4 informational findings were identified.

6.1. `unlockTimeInSec_ ` calculation can be abstracted

Severity: Informational Status: Resolved

Context

• Staking.sol#L304

Description

unlockTimeInSec_ is calculated multiple times for multiple deposits and buys for the same _lockPeriodInSec.

Recommendation

Consider abstracting the calculation to an internal function so it needs to be performed only once.

Moxie

Fixed in fdc31342c088bba72a28498584493d8a3c560f11, d048c5cb6814eb9dca2c55bf475bb8d60a22b60f & 6b1639ef8fd0f6c057f3cd89461b898080939df3.

C4

Fix reviewed



6.2. Add tests for duplicate indexes

Severity: Informational Status: Resolved

Context

- Staking.sol#L416
- Staking.sol#L479-L495

Description

Duplicate indexes are not explictly checked in _extractExpiredAndDeleteLocks, but the function will revert in this scenario with Staking_SubjectsDoesNotMatch because the lock is deleted after it's iterated upon.

This is also the case for the getter getTotalStakedAmount(), but is less of a concern because it is not state-changing.

Recommendation

Add tests for duplicate indexes to ensure that any implementation changes will catch this behaviour.

Moxie

Fixed in commit.

C4

Fix reviewed

6.3. Redundancies

Severity: Informational Status: Resolved

Context

- IStaking.sol#L11
- IStaking.sol#L13
- Staking.sol#L11
- Staking.sol#L18

Description

- The errors Staking_InvalidIndex and Staking_AlreadyWithdrawn are defined but unused.
- OwnableUpgradeable is imported but unused.
- The non-reentrant modifier was removed for all external functions, but the inherited ReentrancyGuard wasn't.

Recommendation

Remove the redundancies.

Moxie

Fixed in <u>27fc92603b294e4641cbb23d737bd6da64700646</u> & bd966bfaa0606347b619e548b85467b3dff29912.

C4

Fix reviewed

6.4. Share buyer isn't correctly logged for `buyAndLockFor()` and `buyAndLockMultipleFor()`

Severity: Informational Status: Resolved

Context

- IStaking.sol#L33-L42
- Staking.sol#L343
- Staking.sol#L406

Description

When buying shares and locking for a beneficiary, the buyer is msg.sender and the recipient is the beneficiary. The Lock event records when the lock was a purchase from the _isBuy parameter, but the _user recorded will be the beneficiary when the purchase was initiated by the caller.

Recommendation

Consider changing isBuy to buyer that logs msg.sender if the lock was created after a buy, address(0) if it isn't.

Moxie

Fixed with the following commit.

C4

Fix reviewed.

