code4rena

Moxie

Smart Contract Security Assessment

Version 2.0

Audit dates: Sep 23 — Sep 24, 2024

Audited by: hickuphh3

Contents

1. Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2. Executive Summary

2.1 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.

2.2 Scope

Repository	moxie-protocol/contracts
Commit Hash	6df8c230f7536b98b96b09c7539a9c855512ee68
Mitigation Hash	6df8c230f7536b98b96b09c7539a9c855512ee68

2.3 Audit Timeline

DATE	EVENT
Sep 23, 2024	Audit start
Sep 24, 2024	Audit end
Oct 23, 2024	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	4
Total Issues	4

3. Findings Summary

ID	DESCRIPTION	STATUS
I-1	Share buyer isn't correctly logged for `buyAndLockFor()` and `buyAndLockMultipleFor()`	Resolved

I-2	Redundancies	Resolved
I-3	Add tests for duplicate indexes	Resolved
I-4	`unlockTimeInSec_ ` calculation can be abstracted	Resolved

4. Findings

4.1 Informational

A total of 4 informational findings were identified.

[I-1] 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

Verified.

[I-2] 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 27fc92603b294e4641cbb23d737bd6da64700646 & bd966bfaa0606347b619e548b85467b3dff29912.

C4 Zenith:

Verified.

[I-3] 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

Verified.

[I-4] `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

Verified.