code4rena

Moxie Findings & Analysis Report

Published: Jul 25, 2024

Prepared for: Moxie

Prepared by: Code4rena

Audit dates: Jul 11—Jul 12, 2024

Contents

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 Jul 11 to Jul 12, 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	1
Informational	0

3. Scope

The source code was delivered to Code4rena in a private 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
Jul 11, 2024	Kick-off call
Jul 11, 2024	Audit start
Jul 12, 2024	Audit end

6. Low Risk Findings (1)

6.1. Delta logic could be sidestepped if address that can pull tokens is added

Severity: Low **Status:** Resolved

Impact

```
        require(_tokenDestinations.add(_dst), "Destination already
added");
```

Risk of sidestep if a contract that can pull tokens is added

Explanation

Since the `fallback` uses a delta balances

```
    uint256 diff = oldBalance.sub(newBalance);
    usedAmount = usedAmount.add(diff);
```

Pulling tokens that are approved will completely sidestep this mechanism

And will allow moving all tokens before they are vested

NOTE

This will be safe when using `MoxieBondingCurve` and `SubjectFactory` and `EasyAuction` as the only `_tokenDestinations`

Additional instances

The tokenManager could also offer the same vector

```
        address subjectToken = tokenManager.tokens(_subject);
```

Moxie: Acknowledged

C4 Pro League: Informational