0xCommit

Security Assessment Report

Metrom

Smart Contracts Audit

Version: Final

Date: 15 Jul 2024

Table of Contents

Table of Contents 1
License 2
Disclaimer 3
Introduction 4
Codebases Submitted for the Audit 5
How to Read This Report 6
Overview 7
Methodology 7
Functionality Overview 7
Summary of Findings 8
Detailed Findings 9
1. Redundant revert should be refactored to modifier or function. 10
2. Unnecessary Assignment of State Variables to Local Variables in createCampaigns Function. 11
3. Unnecessary Inner Loop for Duplicate Check in setMinimumRewardTokenRates Function. 12

License

Disclaimer

Introduction

Purpose of this report

0xCommit has been engaged by **Metrom** to perform a security audit of several Token contract components.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behaviour.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebases Submitted for the Audit

The audit has been performed on the following commits:

Github Link: https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol

Version	Commit hash
Initial	3122502f0a38203d1e9d9943e73798250f9f5aa3
Final	

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
High	An attacker can successfully execute an attack that clearly results in operational issues for the service. This also includes any value loss of unclaimed funds permanently or temporary.
Medium	The service may be susceptible to an attacker carrying out an unintentional action, which could potentially disrupt its operation. Nonetheless, certain limitations exist that make it difficult for the attack to be successful.
Low	The service may be vulnerable to an attacker executing an unintended action, but the impact of the action is negligible or the likelihood of the attack succeeding is very low and there is no loss of value.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary

The status of an issue can be one of the following: **Pending**, **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Overview

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
    d. Access Control Issues
    e. Boundary Analysis
4. Report preparation

Functionality Overview

Metrom is an intuitive tool designed to streamline the incentivization of concentrated liquidity automated market makers (CL AMMs). Building on the concept of Carrot and our experience with DIY Liquidity Campaigns, Metrom simplifies the process into three easy steps. It allows users to distribute rewards to liquidity providers on any trading pair with any reward token. Post-launch, Metrom will enhance this by adding KPI-based conditions to incentives. By reducing friction in campaign creation and focusing on efficiency, Metrom empowers communities to attract more liquidity effectively.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Redundant revert should be refactored to modifier or function.	Informational	Acknowledged
2	Unnecessary Assignment of State Variables to Local Variables in createCampaigns Function.	Informational	Acknowledged
3	Unnecessary Inner Loop for Duplicate Check in setMinimumRewardTokenRates Function.	Informational	Acknowledged

Detailed Findings

Critical Findings

No critical findings found.

High Findings

No high findings found.

Medium Findings

No medium findings found.

Low Findings

No low findings found.

Informational Findings

1. Redundant revert should be refactored to modifier or function.

Description

Refactoring this repeated code into a modifier or function will help reduce deployment costs and enhance code readability.

The codebase contains 10 instances of the following redundant code:

if (msg.sender != owner) revert Forbidden();

1. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L100
2. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L106
3. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L310
4. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L358
5. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L373
6. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L381
7. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L390
8. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L399
9. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L407
10. https://github.com/metrom-xyz/contracts/blob/main/src/Metrom.sol#L415

Remediation

Convert the statement **if (msg.sender != owner) revert Forbidden();** into a reusable function or modifier and apply it wherever necessary. This will improve the efficiency and maintainability of the code.

Status

Acknowledged

2. Unnecessary Assignment of State Variables to Local Variables in createCampaigns Function.

Description

In the createCampaigns function, the state variables **minimumCampaignDuration** and **maximumCampaignDuration** are assigned to local variables **_minimumCampaignDuration** and **_maximumCampaignDuration**, respectively. These local variables are then used only once within the function. This results in redundant code, which can be streamlined for better readability, efficiency and saves gas.

Remediation

Remove the unnecessary assignments of state variables to local variables and use the state variables directly within the function. This will simplify the code and eliminate the redundant variable assignments.

Status

Acknowledged

3. Unnecessary Inner Loop for Duplicate Check in setMinimumRewardTokenRates Function.

Description

The setMinimumRewardTokenRates function currently includes an inner loop to check for duplicate tokens in the _bundles array. This check is redundant and inefficient, especially since the function can only be called by ratesUpdater, who is trusted to pass unique elements. The inner loop unnecessarily increases the gas consumption of the function, leading to higher transaction costs.

Remediation

Remove the inner loop that checks for duplicate tokens within the _bundles array. By trusting the ratesUpdater to provide unique elements, the function's gas consumption can be significantly reduced.

Status

Acknowledged