certora

Security Assessment
Final Report

Metrom

August 2025

Prepared for Metrom

certora

Table of Contents

certora

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
Metrom	https://github.com/metrom-xyz/aptos-contracts	Initial 089bbd4 Fix dcb0c65 Fix 98b4302	Aptos

Project Overview

This document describes the verification of **Metrom** using manual code review. The work was undertaken from **August 11th** to **August 18th, 2025**

The following contract list is included in our scope:

```
sources/metrom.move
sources/helper.move
```

During the manual audit, the Certora team discovered bugs in the Move contracts code, as listed on the following page.

Protocol Overview

Metrom is a flexible liquidity mining platform designed to help DEXes and projects efficiently launch and manage multiple incentivization campaigns. With a focus on security, ease of use, and efficiency, Metrom enables projects to create campaigns quickly, with minimal leakage, while optimizing rewards for liquidity providers.

certora

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	1	1	1
Medium	4	1	1
Low	2	1	1
Informational	8	4	4
Total	15	7	7

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

certora

Detailed Findings

ID	Title	Severity	Status
H-01	Campaign's owner is not enforced in the recover_rewards function	High	Fixed
M-01	Incorrect logic in accept_campaign_ownership	Medium	Fixed
M-02	Campaign's beginning and ending are not accounted for	Medium	Acknowledged
M-03	Initialization can be performed by an attacker	Medium	Acknowledged
M-04	Without root, funds are locked	Medium	Acknowledged
L-01	Updater can steal all funds	Low	Acknowledged
L-02	Fee is calculated using amount instead of received_amount	Low	Fixed
I-01	CreatePointsCampaign event not emitted	Informational	Fixed
I-02	ID collision between rewards and points campaigns	Informational	Acknowledged
I-03	Incorrect variable name	Informational	Acknowledged
I-04	Caller is not included in the reward/point campaign id computation	Informational	Fixed

I-05	Module initialization function visibility can be reduced	Informational	Fixed
I-06	No maximum future campaign start time	Informational	Acknowledged
I-07	Multiple typos across the documentation	Informational	Fixed

High Severity Issues

H-01 Campaign's owner is not enforced in the recover_rewards function

Severity: **High**	Impact: **High**	Likelihood: **Medium**
Files: metrom.move#L949-L967	Status: Fixed	

Description: Anybody can successfully call `recover_rewards` and specify a receiver they control in case they have possession of the proof. This is the case because the `process_multiple_claims` function does not change the `process_reward_claim`'s parameter `enforce_campaign_owner` to `true` when the recovering parameter is also `true`.

JavaScript

```
    fun process_multiple_claims(
        caller: &signer,
        campaign_ids: vector<vector<u8>>,
        proofs: vector<vector<vector<u8>>>,
        tokens: vector<address>,
        amounts: vector<u64>,
        receivers: vector<address>,
        recovering: bool
    ) acquires State {
        let campaign_ids_len = campaign_ids.length();
        assert!(
            proofs.length() == campaign_ids_len
                && tokens.length() == campaign_ids_len
                && amounts.length() == campaign_ids_len
                && receivers.length() == campaign_ids_len,
```

```
            EInconsistentArrayLengths
        );

        let caller_address = signer::address_of(caller);

        for (i in 0..campaign_ids_len) {
            let campaign_id = campaign_ids[i];
            let token = tokens[i];
            let amount = amounts[i];
            let proof = proofs[i];
            let receiver = receivers[i];

            let amount =
                process_reward_claim(
                    caller_address,
                    campaign_id,
                    if (recovering)@0x0
                    else caller_address, // claim_owner
@>                  false, // enforce_campaign_owner
                    proof,
                    token,
                    amount,
                    receiver
                );

            if (recovering) event::emit(
                RecoverReward { campaign_id, token, amount, receiver }
            )
            else event::emit(
                ClaimReward { campaign_id, token, amount, receiver }
            )
        }
```

```
    }
```

Exploit Scenario: An attacker who is in possession of the recovery proof can call `recover_rewards` and steal the funds.

Recommendations: Consider introducing a check for the `enforce_campaign_owner` parameter to be `true` if recovering is also `true` to make sure only the campaign owner can recover the funds.

Customer's response: Fixed in commit dcb0c65.

Fix Review: Fix confirmed.

Medium Severity Issues

M-01 Incorrect logic in accept_campaign_ownership

Severity: **Medium**	Impact: **Low**	Likelihood: **High**
Files: metrom.move#L1032	Status: Fixed	

Description: Point campaign ownership transfer is not possible since `accept_campaign_ownership` checks whether the caller is the current owner instead of checking if the caller is the pending one.

Rust

```
    public entry fun accept_campaign_ownership(
        caller: &signer, id: vector<u8>
    ) acquires State {
        let caller_address = signer::address_of(caller);
        let state = borrow_mut_state();

        if (state.rewards_campaign.contains(id)) {
            let rewards_campaign =
state.rewards_campaign.borrow_mut(id);
            assert!(
  @>
rewards_campaign.pending_owner.contains(&caller_address), EForbidden
            );
            rewards_campaign.owner = caller_address;
            rewards_campaign.pending_owner = option::none();
        } else if (state.points_campaign.contains(id)) {
```

```
            let points_campaign =
state.points_campaign.borrow_mut(id);
  @>        assert!(points_campaign.owner == caller_address,
EForbidden);
            points_campaign.owner = caller_address;
            points_campaign.pending_owner = option::none();
        } else {
            assert!(false, ENonExistentCampaign);
        };

        event::emit(AcceptCampaignOwnership { campaign_id: id, owner:
caller_address });
    }
```

Exploit Scenario: In case the owner of a points campaign transfers the ownership to another user, and this user calls `accept_campaign_ownership`, the call will revert since the function will incorrectly check if the caller is the current owner. The transfer of the points campaign's ownership becomes impossible.

Recommendations: Consider editing the check to compare the caller with the pending owner.

Customer's response: Fixed in commit 98b4302.

Fix Review: Fix confirmed.

M-02 Campaign's beginning and ending are not accounted for

Severity: **Medium**	Impact: **Low**	Likelihood: **High**
Files: metrom.move	Status: Acknowledged	

Description: Campaign `from` and `to` parameters are only validated to be within allowed limits upon campaign creation. However, these values are not used in `claim_rewards` or `recover_rewards` to limit users to successfully call `claim_rewards` only during the campaign period, and the campaign owner to be able to call `recover_rewards` only after the campaign has finished.

Exploit Scenario: The campaign creator can recover rewards during the active campaign period, and users can claim rewards even after the campaign has finished.

Recommendations: Consider introducing checks in both `claim_rewards` and `recover_rewards` that will allow successful calls only during appropriate conditions.
For instance, allow users to claim their rewards **starting** at `from` and **ending** at `to + 1 month`.
For a recovery scenario, allow campaign owners to clawback the rewards **starting** at `to + 1 month`.

Customer's response: Acknowledged. This is currently how we are working, also on EVM chains. The updater takes care to set a Merkle root for a Merkle tree that allows both user claims and campaign owner reimbursements to be done "concurrently", even while the campaign is running, and after, indefinitely.

M-03 Initialization can be performed by an attacker

Severity: **Medium**	Impact: **High**	Likelihood: **Low**
Files: metrom.move#L466	Status: Acknowledged	

Description: The `init_state` function is callable by anybody.

Exploit Scenario: An attacker can initialize the module with arbitrary values, making himself the owner.

Recommendations: Consider calling `init_state` and `init_module` atomically in the deployment scripts or implement access control in the `init_state` function to only allow a trusted address to execute it.
For instance, you can add an entry under the `addresses` section in `Move.toml` and only allow this address to execute the function.

Customer's response: Acknowledged. Since we implement an anti-reinitialization check the only thing to be careful about is to be able to initialize the contract correctly at deployment time

M-04 Without root, funds are locked

Severity: **Medium**	Impact: **High**	Likelihood: **Low**
Files: metrom.move	Status: Acknowledged	

Description: The updater is responsible for updating the root of a campaign. In case the updater does not update the root for whatever reason, the owner of the campaign will not be able to retrieve the funds, effectively locking them inside the module.

Recommendations: Consider introducing a function that will allow the campaign owner to recover the funds in case the root is `0x0` and a certain amount of time has passed.

Customer's response: Acknowledged. In this scenario we could craft an ad-hoc Merkle tree and push its root on-chain to unstuck the funds.

Low Severity Issues

L-01 Updater can steal all funds		
Severity: **Low**	Impact: **High**	Likelihood: **Very low**
Files: metrom.move#L691-L694	Status: Acknowledged	

Description: A compromised `updater` can fake Merkle roots for all the rewards campaigns and subsequently "claim" all the available rewards, robbing the campaign owners of their assets.

Recommendations: Include a signature for every updated root in `distribute_rewards`, which must be signed by the respective campaign owner.

Customer's response: Acknowledged.

L-02 Fee is calculated using amount instead of received_amount

Severity: **Low**	Impact: **Very low**	Likelihood: **High**
Files: metrom.move#L568	Status: Fixed	

Description: In `create_rewards_campaign`, the function can receive more tokens than required but only calculates fees based on the required amount, not the actual received amount.

The `fee_amount` variable calculates fee using `amount`, not `received_amount`, and right after, the `reward_amount_minus_fees` uses `received_amount`, not `amount` again, which is inconsistent.

Recommendations: We recommend calculating fees based on `received_amount` instead of `amount`.

Customer's response: Fixed in commit 98b4302.

Fix Review: Fix confirmed.

Informational Issues

I-01. CreatePointsCampaign event not emitted

Description: Though the `CreatePointsCampaign` is defined in the module, it is never emitted

Recommendation: Emit `CreatePointsCampaign` at the end of `create_points_campaign`.

Customer's response: Fixed in commit 98b4302.

Fix Review: Fix confirmed.

I-02. ID collision between rewards and points campaigns

Description: If there are two campaigns (one rewards and one points) with the same `id`, the ownership of the points campaign can't be transferred (using `transfer_campaign_ownership`). Because of the differences between the `RewardsCampaign` and `PointsCampaign` structures, this seems to be impossible to achieve in practice.

Recommendation: Adding a different identifier string at the start or end of the `RewardsCampaign` and `PointsCampaign` structures, or creating a different function for ownership transfer for every campaign type.

Customer's response: Due to extremely low likelihood, the issue won't be addressed.

I-03. Incorrect variable name

Description: The state variable `minimum_fee_token_rate` is not really a minimum, but rather the actual fee token's rate taken by the protocol.

Recommendation: Rename the variable.

Customer's response: The issue won't be acted on.

I-04. Caller is not included in the reward/point campaign id computation

Description: Anyone could copy the parameters of `create_points_campaign` or `create_rewards_campaign` and front-run the original call. This would result in the original call being reverted.

Since the creator has to pay for the rewards and the fees, it is not a significant issue but by including the caller in the id generation, there could be multiple campaigns with the same parameters at the same time.

Recommendation: Include the caller address in the campaign id computation.

Customer's response: Fixed in commit 98b4302.

Fix Review: Fix confirmed.

I-05. Module initialization function visibility can be reduced

Description: The `init_module` function is the constructor of the module and is run only once when the package is published. It is not meant to be executed again so its visibility can be restricted to private.

Recommendation: Update the function signature like such: `fun init_module(caller: &signer).`

Customer's response: Fixed in commit 98b4302.

Fix Review: Fix confirmed.

I-06. No maximum future campaign start time

Description: Campaigns can be created with start times arbitrarily far in the future, for example in 1000 years, locking funds indefinitely.

Recommendation: We recommend adding a maximum future start time limit, for example 1 year from current time.

Customer's response: The issue won't be acted on.

I-07. Multiple typos across the documentation

Description: During the review multiple types and wrong documentation parts were spotted. These are:

- `CreateRewardsCampaign` documentation
  - `@param A list of the fees paid to create the campaign.`
    → `@param reward_fees A list of the fees paid to create the campaign.`
- `DistributeReward` documentation
  - `The id of the campaign. on which the rewards were distributed.`
    → `The id of the campaign on which the rewards were distributed.`
- `ClaimFee` documentation
  - `The claims's receiver.`
    → `The claim's receiver.`
- `TransferCampaignOwnership` event
  - `@param id The targeted`
    → `@param campaign_id The id of the targeted campaign.`
- `AcceptCampaignOwnership` event
  - `The targete campaign's id.`
    → `The targeted campaign's id.`
  - `The targete campaign's new owner.`
    → `The targeted campaign's new owner.`
  - Also param name: `@param id`
    → `@param campaign_id …`
- `accept_campaign_ownership` documentation

    - `Finalized an ownership transfer operation…`
      → `Finalizes an ownership transfer operation…`
- `create_rewards_campaign` documentation
    - `pointing fo a file`
      → `pointing to a file`
- `create_points_campaign` documentation
    - `pointing fo a file`
      → `pointing to a file`
- `Distribute_rewards` documentation
    - Uses singular params (`campaign_id, root`) but function takes vectors
      → `@param campaign_ids …, @param roots …` and pluralize text accordingly - `set Merkle roots for the campaigns`.
- `set_minimum_token_rates` documentation
    - `Sets the minimum rate for an allowed reward token`.
      → `Sets minimum rates for allowed reward and fee tokens`
- `claim_fees` documentation
    - Duplicate param name:
        - `@param token The token to claim.`
        - `@param token The receiver of the claim.`
          → The second line should be `@param receiver The receiver of the claim.`

Recommendation: We recommend correcting the documentation and comments accordingly.

Customer's response: Fixed in commit [98b4302](98b4302).

Fix Review: Fix confirmed.

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.