SHERLOCK

Security Review For Metrom

Collaborative Audit Prepared For: **Metrom**
Lead Security Expert(s): **ctf_sec**
deth
Date Audited: **January 8 - January 12, 2026**

Introduction

Metrom (https://x.com/metromxyz) is an incentive orchestration platform built to design, launch, and manage on-chain incentive programs. It began with a focus on concentrated AMMs and is now expanding to support a wider range of DeFi primitives including CDPs, lending and borrowing. Metrom helps protocols optimize capital attraction and retention through performance-driven campaigns.

Scope

Repository: metrom-xyz/contracts

Audited Commit: a6181120b7f952d20bae5956d9f0972f4e64a6db

Final Commit: a6181120b7f952d20bae5956d9f0972f4e64a6db

Files:

- src/IMetrom.sol
- src/libraries/BaseCampaignsUtils.sol
- src/libraries/PointsCampaignsV1Utils.sol
- src/libraries/PointsCampaignsV2Utils.sol
- src/libraries/RewardsCampaignsV1Utils.sol
- src/libraries/RewardsCampaignsV2Utils.sol
- src/Metrom.sol

Final Commit Hash

a6181120b7f952d20bae5956d9f0972f4e64a6db

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	1

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: `collectPointsCampaignFee` does not support fee-on-transfer [ACKNWOLEDGED]

Source: https://github.com/sherlock-audit/2026-01-metrom-jan-8th/issues/8

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

While the protocol implements logic to support Fee-on-Transfer (FoT) tokens in `RewardsCampaigns` (by calculating balance differences),

however, this support is missing in the `collectPointsCampaignFee` function used for `PointsCampaigns`. If a Fee-on-Transfer token is whitelisted as a fee token, any attempt to create a points campaign using it will inevitably revert.

Vulnerability Detail

The collectPointsCampaignFee function calculates the actual amount received by the contract (`_collectedFeeAmount`) by checking the balance difference before and after the transfer.

https://github.com/sherlock-audit/2026-01-metrom-jan-8th/blob/be359a3a9428df71da86bc597143c93433f5ea68/contracts/src/Metrom.sol#L324

However, it immediately enforces a strict check requiring the collected amount to be at least equal to the `_requiredFeeAmount` (the amount requested in `safeTransferFrom`).

For tokens that implement a transfer tax (Fee-on-Transfer), the actual amount received by the recipient is always strictly less than the amount sent.

1. `safeTransferFrom` sends `X`.
2. The token contract takes a tax `T`.
3. The contract receives `X - T`.
4. The check `(X - T) < X` evaluates to `true`.
5. The transaction reverts with `FeeAmountTooLow()`.

Impact

Denial of Service for Supported Assets:

The protocol explicitly aims to support Fee-on-Transfer tokens (as evidenced by the logic in `createRewardsCampaign` and previous audit fixes). However, users are completely blocked from using these tokens to pay for Points Campaigns, even if the token is whitelisted by the protocol owner.

Code Snippet

File: `Metrom.sol` **Function:** collectPointsCampaignFee

```
function collectPointsCampaignFee(address _feeToken, uint256 _requiredFeeAmount)
 ↪  internal returns (uint256) {
    uint256 _balanceBefore = IERC20(_feeToken).balanceOf(address(this));

    // Transfer strictly _requiredFeeAmount
    IERC20(_feeToken).safeTransferFrom(msg.sender, address(this),
     ↪  _requiredFeeAmount);

    // Calculate actual received amount
    uint256 _collectedFeeAmount = IERC20(_feeToken).balanceOf(address(this)) -
     ↪  _balanceBefore;

    // @audit-issue Strict check causes revert for FoT tokens because
     ↪  _collectedFeeAmount will be < _requiredFeeAmount
    if (_collectedFeeAmount < _requiredFeeAmount) revert FeeAmountTooLow();

    claimableFees[_feeToken] += _collectedFeeAmount;
    return _collectedFeeAmount;
}
```

Tool Used

Manual Review

Recommendation

Update the `collectPointsCampaignFee` logic to allow the collected fee to be lower than the transfer amount (implying the protocol accepts the "net" amount after tax), or update the parameters to allow a tolerance for the tax.

If the protocol intends to accept the post-tax amount as full payment, the strict inequality check should be modified.

Proposed Fix: Update the function to accept an `expectedAmount` (the minimum acceptable net amount) or simply remove the strict equality check if any non-zero amount is acceptable.

```
- function collectPointsCampaignFee(address _feeToken, uint256 _requiredFeeAmount)
 ↪  internal returns (uint256) {
+ function collectPointsCampaignFee(address _feeToken, uint256 _requiredFeeAmount,
 ↪  uint256 minExpectedAmount) internal returns (uint256) {
      uint256 _balanceBefore = IERC20(_feeToken).balanceOf(address(this));
```

```
      IERC20(_feeToken).safeTransferFrom(msg.sender, address(this),
      ↪  _requiredFeeAmount);
      uint256 _collectedFeeAmount = IERC20(_feeToken).balanceOf(address(this)) -
      ↪  _balanceBefore;

-     if (_collectedFeeAmount < _requiredFeeAmount) revert FeeAmountTooLow();
+     if (_collectedFeeAmount < minExpectedAmount) revert FeeAmountTooLow();

      claimableFees[_feeToken] += _collectedFeeAmount;
      return _collectedFeeAmount;
  }
```

Discussion

luzzif

While this is valid and acknowledged, we do whitelist our fee tokens so we will make sure to never add fee on transfer tokens to point campaigns.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.