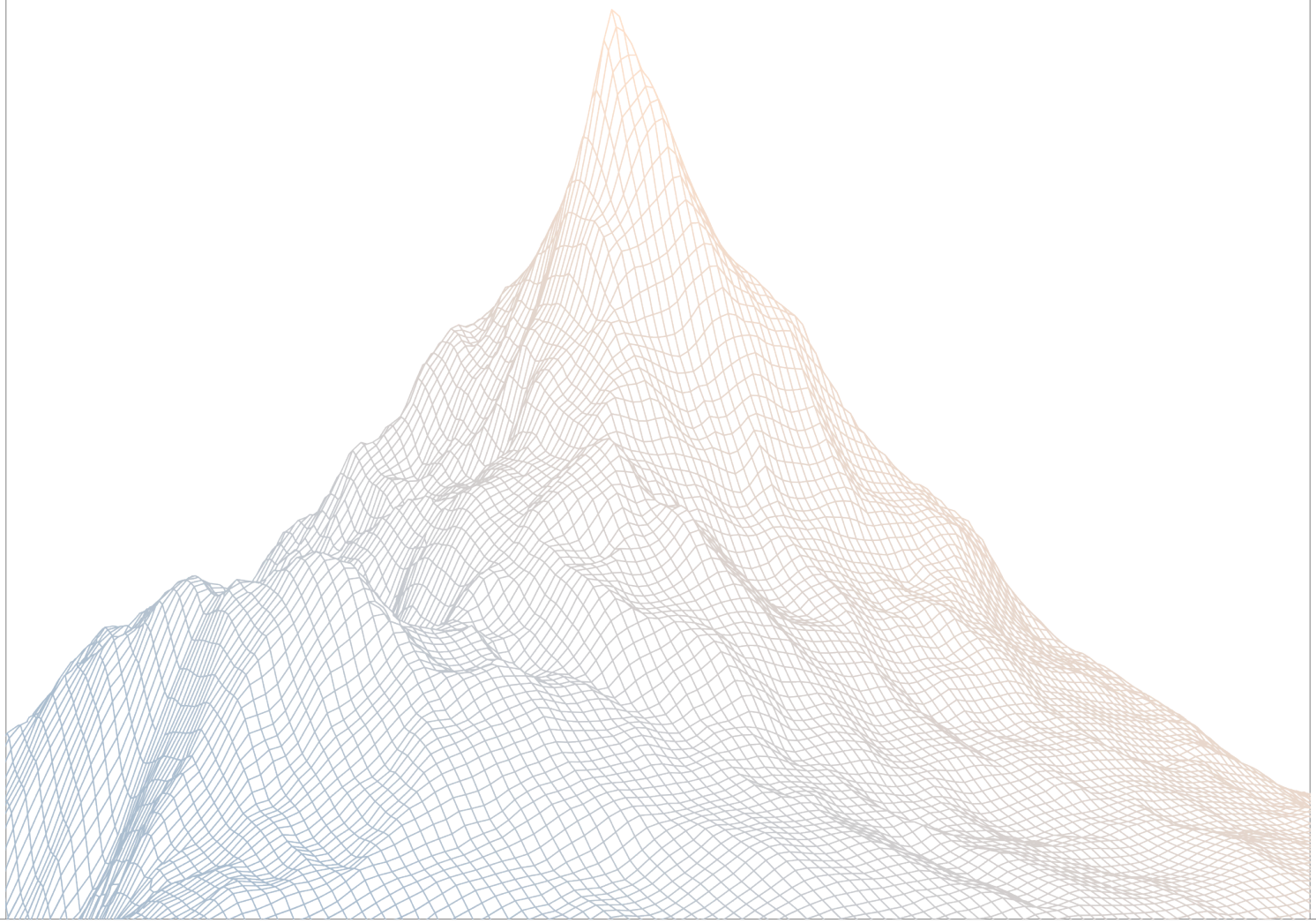
Zenith

Meteora

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: October 8th to October 10th, 2025

AUDITED BY: J4X
peakbolt

Contents

1 Introduction

1.1 About Zenith

Zenith assembles auditors with proven track records: finding critical vulnerabilities in public audit competitions.

Our audits are carried out by a curated team of the industry's top-performing security researchers, selected for your specific codebase, security needs, and budget.

Learn more about us at https://zenith.security.

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 Meteora

The Dynamic Fee Sharing program allows for the creation of fee vaults where collected fees can be automatically distributed among multiple recipients based on predetermined share allocations.

2.2 Scope

The engagement involved a review of the following targets:

Target	dynamic-fee-sharing
Repository	https://github.com/MeteoraAg/dynamic-fee-sharing
Commit Hash	`a1e79d65851a08ec9f941c70e08c410f1cfdb8e4`
Files	`Changes in PR #8`

2.3 Audit Timeline

October 8, 2025	Audit start
October 10, 2025	Audit end
October 15, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	2
Low Risk	4
Informational	1
Total Issues	**7**

3

Findings Summary

ID	Description	Status
M-1	DBC surplus claims can revert if curve has not finished yet	Resolved
M-2	Missing restriction on claims allow for grieving	Resolved
L-1	Fee sharing can't be used with memo required receivers	Acknowledged
L-2	No IX for claim_rewards for funding by DAMM-v2	Resolved
L-3	Missing withdraw_migration_fee IX for funding by DBC	Resolved
L-4	funding_by_claim_dbc_creator_trading_fee fails to verify config account against pool	Acknowledged
I-1	Incorrect error msg for InvalidDbcPool	Resolved

4

Findings

4.1 Medium Risk

A total of 2 medium risk findings were identified.

[M-1] DBC surplus claims can revert if curve has not finished yet

SEVERITY: Medium	IMPACT: Medium
STATUS: Resolved	LIKELIHOOD: Medium

Target

- /src/instructions/ix_funding_by_claim_dbc_creator_surplus.rs#L52
- /src/instructions/ix_funding_by_claim_dbc_partner_surplus.rs#L52

Description:

The dynamic fee-sharing claim functions are designed to never revert. This is implemented to ensure that integrations built on top and potentially looping over multiple claims don't fail. To ensure this, the functions `handle_funding_by_claim_dbc_creator_surplus` and `handle_funding_by_claim_dbc_partner_surplus` return ok if the flags that represent that the surplus has already been claimed are set.

```
if virtual_pool.is_partner_withdraw_surplus == 1 {
    return Ok(());
}
```

```
if virtual_pool.is_creator_withdraw_surplus == 1 {
    return Ok(());
}
```

However, this forgets that there is another reason why the surplus claims can revert. This is if the curve has not finished yet.

```
require!(
    pool.is_curve_complete(config.migration_quote_threshold),
    PoolError::NotPermitToDoThisAction
```

```
);
```

As a result, integrations will still fail if they loop over multiple claims and the curve hasn't finished yet for one of them.

Recommendations:

We recommend also returning `ok()` if `!pool.is_curve_complete(config.migration_quote_threshold)`.

Meteora: For our new changes, we wont return OK(()) in those claiming fees function, instead it replies totally on CPI. Resolved with PR-11.

Zenith: Verified. Mitigated by removing the feature

[M-2] Missing restriction on claims allow for grieving

SEVERITY: Medium	IMPACT: Medium
STATUS: Resolved	LIKELIHOOD: Medium

Target

- src/instructions/ix_funding_by_claim_damm_v2.rs#L14
- src/instructions/ix_funding_by_claim_dbc_creator_surplus.rs#L14
- src/instructions/ix_funding_by_claim_dbc_creator_trading_fee.rs#L14
- src/instructions/ix_funding_by_claim_dbc_partner_surplus.rs#L14
- src/instructions/ix_funding_by_claim_dbc_partner_trading_fee.rs#L14

Description:

The newly added claim functions enable anyone to initiate a claim on behalf of the vault. This leads to an issue if the vault is intended for a FOT token with a maximum fee implemented. In that case, vault owners would benefit from calling the claim less frequently to reduce their fee costs. However, in the current implementation, anyone can call the claim more regularly so that the owners always incur the full percentage of the transfer fee.

Recommendations:

We recommend ensuring that only shareholders of the fee vault are authorized to call the functions on behalf of the vault.

Meteora: Resolved with PR-11

Zenith: Verified.

4.2 Low Risk

A total of 4 low risk findings were identified.

[L-1] Fee sharing can't be used with memo required receivers

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- src/instructions/ix_claim_fee.rs#L39-L46

Description:

The `claim_fee` IX is used by users to claim their share of the fees inside a fee vault. However, this claim can err in one edge case. The Token2022 accounts can add an extension that requires every transfer that they receive to include a memo instruction (similar to a traditional bank transfer). However, the current system does not check for this and never adds a memo instruction.

```
let instruction = spl_token_2022::instruction::transfer_checked(
    token_program.key,
    &token_vault.key(),
    &token_mint.key(),
    &token_owner_account.key(),
    &pool_authority.key(),
    &[],
    amount,
    token_mint.decimals,
)?;

let account_infos = vec![
    token_vault.to_account_info(),
    token_mint.to_account_info(),
    token_owner_account.to_account_info(),
    pool_authority.to_account_info(),
];

invoke_signed(&instruction, &account_infos, &[&signer_seeds[..]])?;
```

```
Ok(())
```

As a result, a user account that has the memo extension enabled won't be able to claim their fees.

Recommendations:

We recommend checking if the receiving account has the memo transfer extension added. If that is the case, the program should add a memo instruction to enable claiming.

Meteora: Acknowledged

[L-2] No IX for `claim_rewards` for funding by DAMM-v2

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- lib.rs

Description:

The DAMM-V2 allows the position owner to claim rewards earned by the position.

However, the `dynamic_fee_sharing` does not have a `claim_rewards()` IX for DAMM-V2 pools. That means any earned rewards by the position would be left in the pool and unclaimable.

Recommendations:

Consider adding a `claim_rewards()` for DAMM-V2 pools.

Meteora: Resolved with PR-11 and PR-13.

Zenith: Resolved by allowing claim rewards IX to be called via new funding IX. Note that this only allows reward claim that matches token mint of fee vault.

[L-3] Missing `withdraw_migration_fee` IX for funding by DBC

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- lib.rs

Description:

The `dynamic_fee_sharing` program is missing the IX for `withdraw_migration_fee`.

This means that the program would not be able to withdraw and fund the fee vault with the migration fee if it exists.

Furthermore, the pool's `fee_claimer` and `creator` cannot be changed once its tied to the fee vault, so there is no means to withdraw those migration fees.

Recommendations:

Consider adding a withdraw migration fee funding IX.

Meteora: Resolved with PR-11 and PR-13.

Zenith: Resolved by allowing withdraw migration IX to be called via new funding IX.

[L-4] `funding_by_claim_dbc_creator_trading_fee` fails to verify config account against pool

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- ix_funding_by_claim_dbc_creator_trading_fee.rs#L18-L19
- ix_funding_by_claim_dbc_creator_trading_fee.rs#L66-L69

Description:

The `funding_by_claim_dbc_creator_trading_fee()` IX will verify that the config's collect fee mode is 0 (only quote token).

However, it fails to verify that the `config` account is used by the `pool` account as it only validate based on `quote_mint`.

This will allow users to perform funding from a pool that collect in both tokens, by passing in a config that has `collect_fee_mode == 0`. Though this will not have any significant impact as `claim_creator_trading_fee` CPI will claim zero base amount.

```
pub struct FundingByClaimDbcCreatorTradingFeeCtx<'info> {
    #[account(mut)]
    pub fee_vault: AccountLoader<'info, FeeVault>,

    #[account(has_one = quote_mint)]
    pub config: AccountLoader<'info, PoolConfig>,
```

```
pub fn handle_funding_by_claim_dbc_creator_trading_fee(
    ctx: Context<FundingByClaimDbcCreatorTradingFeeCtx>,
) -> Result<()> {
    let config = ctx.accounts.config.load()?;
    let fee_vault = ctx.accounts.fee_vault.load()?;
    // support collect fee mode is 0 (only quote token)
    require!(config.collect_fee_mode == 0, FeeVaultError::InvalidDbcPool);
```

Recommendations:

This can be resolved by verifying that the `config` account matches `pool.config`.

Meteora: Acknowledged. We wont verify here, otherwise it replies totally on CPI.

Zenith: Acknowledged by client to skip the verification of config account so that the new `fund_by_claiming_fee` IX will more generic and not tied to specific CPI.

4.3 Informational

A total of 1 informational findings were identified.

[I-1] Incorrect error msg for `InvalidDbcPool`

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- error.rs#L35

Description:

The message "Invalid dammv2 pool" is incorrectly used for `InvalidDbcPool`. This will cause anchor to log incorrect error messages.

```
#[msg("Invalid dammv2 pool")]
InvalidDammv2Pool,

#[msg("Invalid dammv2 pool")]
InvalidDbcPool,
```

Meteora: Fixed in PR-11.

Zenith: Resolved by removing the error messages as they are not required for the new `funding_by_claiming_fee` IX.