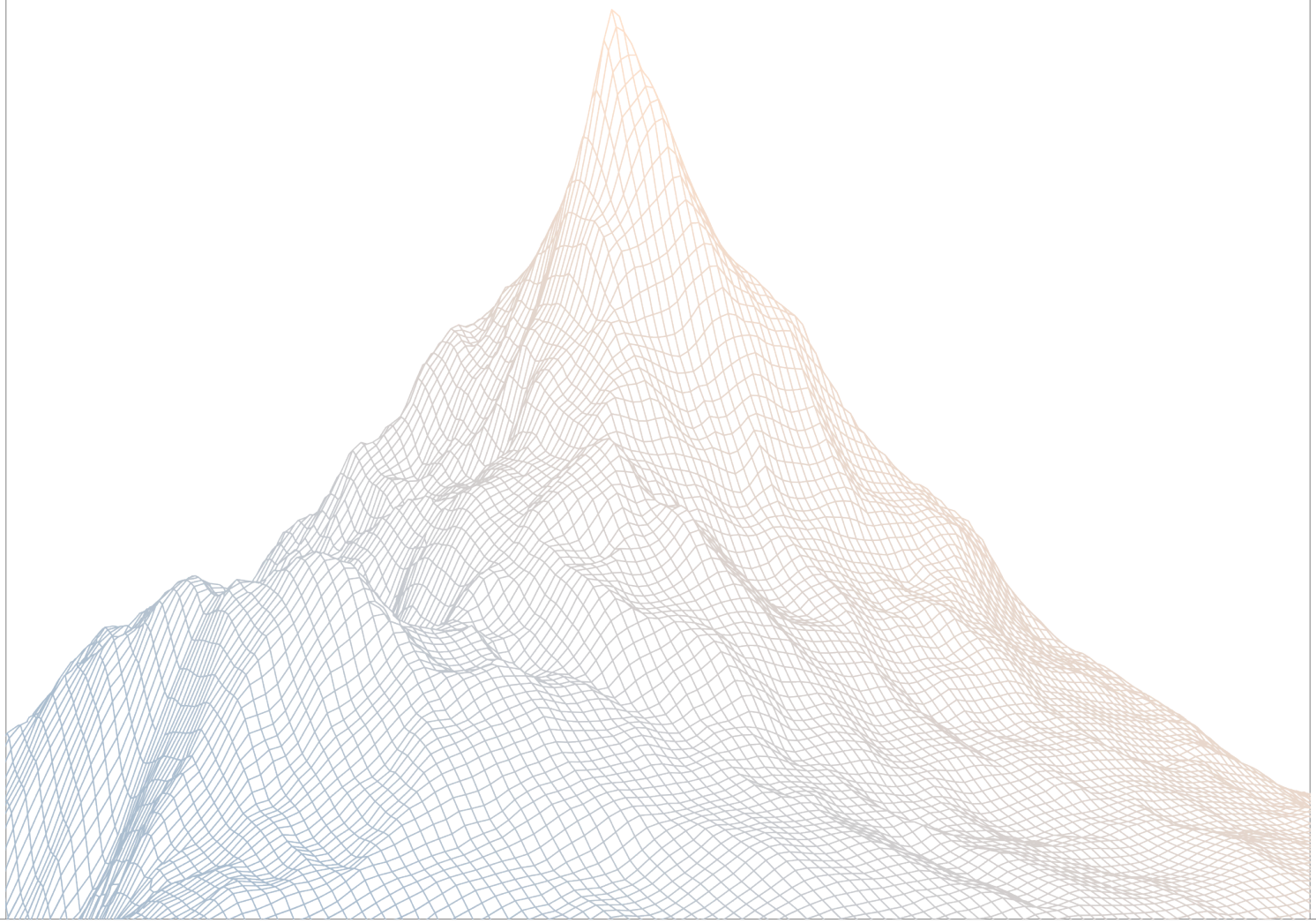
Zenith

Meteora

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: January 26th to January 30th, 2026

AUDITED BY: 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

Our mission is to build the most secure, sustainable and composable liquidity layer for all of Solana and DeFi.

By using Meteora's DLMM and Dynamic AMM Pools, liquidity providers can earn the best fees and yield on their capital.

This would help transform Solana into the ultimate trading hub for mainstream users in crypto by driving sustainable, long-term liquidity to the platform. Join us at Meteora to shape Solana's future as the go-to destination for all crypto participants.

2.2 Scope

The engagement involved a review of the following targets:

Target	dynamic-bonding-curve
Repository	https://github.com/MeteoraAg/dynamic-bonding-curve/pull/165
Commit Hash	`2b1641f6776306ad8b1230ddfbf2f57bbfb8f586`
Files	`Changes in PR-165`

Target	Meteora Mitigation Review
Repository	https://github.com/MeteoraAg/dynamic-bonding-curve/pull/165
Commit Hash	`a5ab5352ad1c597ac11316fd6dfb376b8d53e44e`
Files	`Changes in PR-165`

2.3 Audit Timeline

January 26, 2026	Audit start
January 30, 2026	Audit end
January 30, 2026	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	3
Total Issues	**3**

3

Findings Summary

ID	Description	Status
I-1	Incorrect permissionless doc comment on handle_claim_protocol_fee	Resolved
I-2	Consider adding a rent-exemption check for transfer_lamports_from_pool_account()	Resolved
I-3	Consider always emitting event for claim_protocol_pool_creation_fee to be consistent	Resolved

4

Findings

4.1 Informational

A total of 3 informational findings were identified.

[I-1] Incorrect permissionless doc comment on `handle_claim_protocol_fee`

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

Target

- ix_claim_protocol_fee.rs#L63

Description:

The doc comment for `handle_claim_protocol_fee` incorrect states that it is permissionless, even though it is permissioned.

```
/// Withdraw protocol fees. Permissionless.
pub fn handle_claim_protocol_fee(
    ctx: Context<ClaimProtocolFeesCtx>,
    max_base_amount: u64,
```

Recommendations:

Remove the incorrect doc comment from `handle_claim_protocol_fee`.

Meteora: Resolved with PR-176.

Zenith: Verified.

[I-2] Consider adding a rent-exemption check for `transfer_lamports_from_pool_account()`

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

Target

- token.rs#L164

Description:

`transfer_lamports_from_pool_account()` is intended to transfer lamports out of the pool account, specifically for `claim_protocol_pool_creation_fee` and `claim_partner_pool_creation_fee`.

Currently, there are no issues with it for the specified claiming of pool creation fees as the logic for legacy/current fees are properly defined and fee amount hardcoded.

Though, a safer approach is to add a check to `transfer_lamports_from_pool_account()` to ensure `pool_lamports_after ⩾ Rent::minimum_balance(pool.data_len())`.

This will guard against the scenario where an incorrect change to pool creation fee is made, such that it transfers out more than expected and causes the pool account to no longer be rent exempted, which will have a devastating impact.

```
pub fn transfer_lamports_from_pool_account<'info>(
    pool: AccountInfo<'info>,
    to: AccountInfo<'info>,
    lamports: u64,
) → Result<()> {
    pool.sub_lamports(lamports)?;
    to.add_lamports(lamports)?;

    Ok(())
}
```

Recommendations:

Consider adding a check in `transfer_lamports_from_pool_account()` to ensure `pool_lamports_after ⩾ Rent::minimum_balance(pool.data_len())`.

Meteora: Resolved with PR-176.

Zenith: Verified. Resolved by ensuring `pool` account is still rent exempt after transfer.

[I-3] Consider always emitting event for `claim_protocol_pool_creation_fee` to be consistent

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

Target

- ix_claim_protocol_pool_creation_fee.rs#L58-L62

Description:

`claim_protocol_pool_creation_fee` only emit `EvtClaimPoolCreationFee` when `protocol_fee > 0`.

This is inconsistent with `claim_protocol_fee()` and other claim IXs, which emit events regardless of claim amount.

```
if protocol_fee > 0 {
    transfer_lamports_from_pool_account(
        ctx.accounts.pool.to_account_info(),
        ctx.accounts.treasury.to_account_info(),
        protocol_fee,
    )?;

    emit_cpi!(EvtClaimPoolCreationFee {
        pool: ctx.accounts.pool.key(),
        receiver: ctx.accounts.treasury.key(),
        creation_fee: protocol_fee,
    });
}
```

Recommendations:

For consistency, consider always emitting `EvtClaimPoolCreationFee` in `claim_protocol_pool_creation_fee`.

Meteora: Resolved with PR-176.

Zenith: Verified.