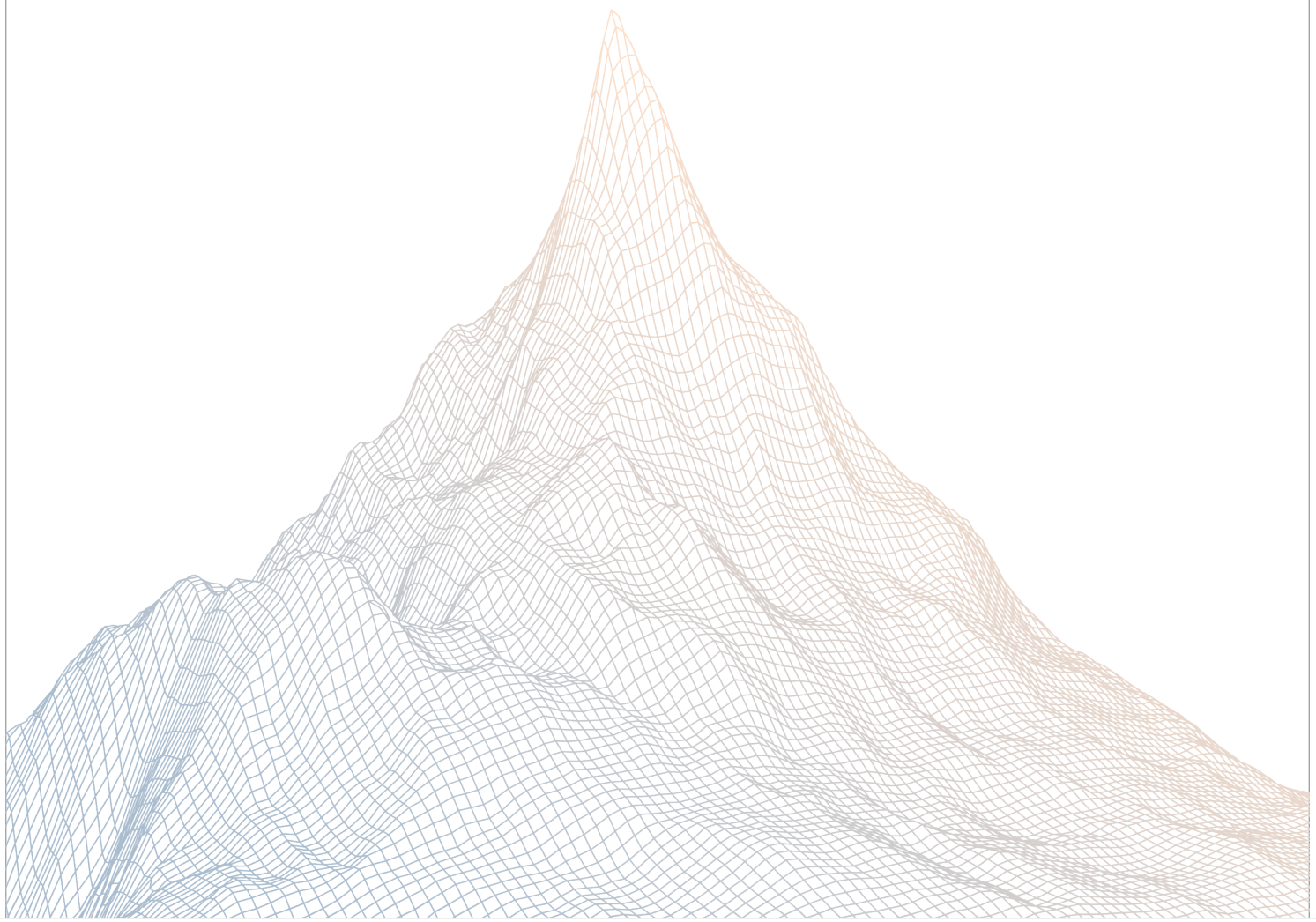
Zenith

Meteora

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: June 19th to June 20th, 2025

AUDITED BY: Mario Poneder
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
Commit Hash	`TODO: 3bf57ac1800938afb32c3fbf206aa6a49ec2e3a3`
Files	`Changes in the PR-83`

Target	damm-v2
Repository	https://github.com/MeteoraAg/damm-v2
Commit Hash	`f038ba6d13f99948270ff94a315874a1df244fe7`
Files	`Changes in the PR-52`

2.3 Audit Timeline

June 19, 2025	Audit start
June 20, 2025	Audit end
June 26, 2025	Report published

2.4 Issues Found

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

3

Findings Summary

ID	Description	Status
M-1	TokenMetadata is always mutable by pool creator	Resolved
I-1	Admin cannot initialize or update reward at index zero	Resolved

4

Findings

4.1 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] `TokenMetadata` is always mutable by pool creator

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

Target

- ix_initialize_virtual_pool_with_token2022.rs#L127-L134

Description:

The `initialize_virtual_pool_with_token2022` instruction will set the metadata's update authority based on the config's token authority settings.

However, it only sets the MetadataPointer extension's update authority and leaves the TokenMetadata extension's update authority as the creator.

This allows the pool creator to update the metadata fields even when the configured update authority is set to partner or immutable.

```
pub fn handle_initialize_virtual_pool_with_token2022<'c: 'info, 'info>(
    ctx: Context<'_, '_, 'c, 'info,
    InitializeVirtualPoolWithToken2022Ctx<'info>>,
    params: InitializePoolParameters,
) → Result<()> {
    let config = ctx.accounts.config.load()?;
    let token_type_value =
        TokenType::try_from(config.token_type).map_err(|_|
    PoolError::InvalidTokenType)?;
    require!(
        token_type_value == TokenType::Token2022,
        PoolError::InvalidTokenType
    );

    let InitializePoolParameters { name, symbol, uri } = params;
```

```
    // initialize metadata
    let cpi_accounts = TokenMetadataInitialize {
        program_id: ctx.accounts.token_program.to_account_info(),
        mint: ctx.accounts.base_mint.to_account_info(),
        metadata: ctx.accounts.base_mint.to_account_info(),
        mint_authority: ctx.accounts.pool_authority.to_account_info(),
        //@audit this is left to creator and not change later
        update_authority: ctx.accounts.creator.to_account_info(),
    };
    let seeds = pool_authority_seeds!(const_pda::pool_authority::BUMP);
    let signer_seeds = &[&seeds[..]];
    let cpi_ctx = CpiContext::new_with_signer(
        ctx.accounts.token_program.to_account_info(),
        cpi_accounts,
        signer_seeds,
    );
    token_metadata_initialize(cpi_ctx, name, symbol, uri)?;
```

POC:

Make the following changes to the respective files and run `create_pool_with_token2022.tests.ts`. it will fail as the token metadata is updated using creator's signature despite using `PartnerUpdateAuthority`.

`create_pool_with_token2022.tests.ts`

```
    it("Partner create config", async () => {
           ...
            //@audit set PartnerUpdateAuthority
           tokenUpdateAuthority: 2,
           tokenUpdateAuthority: 0,
           ...
```

`handle_initialize_virtual_pool_with_token2022.rs`

```
use anchor_spl::{
    token_2022::{mint_to, MintTo, Token2022},
    token_interface::{
        token_metadata_update_field, TokenMetadataUpdateField,
...
pub fn handle_initialize_virtual_pool_with_token2022<'c: 'info, 'info>(
   ...
    emit_cpi!(EvtInitializePool {
```

```
        pool: ctx.accounts.pool.key(),
        config: ctx.accounts.config.key(),
        creator: ctx.accounts.creator.key(),
        base_mint: ctx.accounts.base_mint.key(),
        pool_type: PoolType::Token2022.into(),
        activation_point,
    });

    msg!("updating metadata field....");
    // initialize metadata
    let cpi_accounts = TokenMetadataUpdateField {
        program_id: ctx.accounts.token_program.to_account_info(),
        metadata: ctx.accounts.base_mint.to_account_info(),
        update_authority: ctx.accounts.creator.to_account_info(),
    };
    let cpi_ctx = CpiContext::new(
        ctx.accounts.token_program.to_account_info(),
        cpi_accounts,
    );
    token_metadata_update_field(cpi_ctx, Field::Name, "updated name".to_
        string())?;

    Ok(())
}
```

Recommendations:

Use anchor helper `token_metadata_update_authority()` to set the correct update authority for the TokenMetadata extension.

Meteora: Resolved with PR-98.

Zenith: Verified.

4.2 Informational

A total of 1 informational findings were identified.

[I-1] Admin cannot initialize or update reward at index zero

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

Target

- programs/cp-amm/src/state/pool.rs#L744-L758
- programs/cp-amm/src/instructions/admin/ix_initialize_reward.rs#L62
- programs/cp-amm/src/instructions/admin/ix_update_reward_duration.rs#L44
- programs/cp-amm/src/instructions/admin/ix_update_reward_funder.rs#L25

Description

When a pool has a non-default `creator` set, the admin is unable to initialize the reward at `reward_index == 0`, nor update its reward duration or funder. This is due to the logic in `Pool::validate_authority_to_edit_reward`:

```
pub fn validate_authority_to_edit_reward(
    &self,
    reward_index: usize,
    signer: Pubkey,
) -> Result<()> {
    // legacy pools would have creator's pubkey as Pubkey::default()
    if reward_index == 0 && self.creator.ne(&Pubkey::default()) {
        // only pool creator is allowed to initialize reward with index 0
        require!(self.creator == signer, PoolError::InvalidPoolCreator);
    } else {
        require!(assert_eq_admin(signer), PoolError::InvalidAdmin);
    }
    Ok(())
}
```

This function is called by the `validate` function in the following instructions:

- `initialize_reward`
- `update_reward_duration`
- `update_reward_funder`

Recommendations

It is recommended to consider updating the logic in `validate_authority_to_edit_reward` to allow admins to initialize or update reward at index zero, even when a pool creator is set, if that is the intended behavior. If only the creator should have this authority, it is beneficial to clarify this restriction in the documentation and comments.

Meteora: Resolved with PR-76

Zenith: Verified.