Obsidian

AUDITS

Pump Science Security Review

Auditors

0xjuaan

0xSpearmint

30th May, 2025

Introduction

Obsidian Audits

Obsidian audits is a team of top-ranked security researchers, with a publicly proven track record, specialising in DeFi protocols across EVM chains and Solana.

The team has achieved 10+ top-2 placements in audit competitions, achieving top rankings in competitions for Uniswap v4, Pump.fun, Sentiment, and many more.

Find out more: https://github.com/ObsidianAudits

Audit Overview

Pump Science is a Solana-based protocol for funding scientific research with trading fees.

Pump Science engaged Obsidian Audits to perform a security review on their updated migration instruction, migrating to Meteora's DAMM v2. The review was conducted between the 28th and 29th of May.

Scope

Files in scope

Repo	Files in scope
`moleculeprotocol/pump-science-contract` **Branch:** `damm-v2-migration` **Commit hash:** 7fb0c1ff3dd17b5b5a341ca5571e01daa5aeb882	`src/instructions/migration/migrate_damm_v2.rs`

Summary of Findings

Severity Breakdown

A total of **4** issues were identified and categorized based on severity:

- **2 Medium severity**
- **2 Informational**

Findings Overview

ID	Title	Severity	Status
M-01	`CREATE_POOL_RENT_FEE` is subtracted from the quote token amount unnecessarily	Medium	Fixed
M-02	Migration can fail due to rounding errors when splitting liquidity between the platform and dev	Medium	Fixed
I-01	Unused `associated_token_program` account in MigrateDammV2 account struct	Informational	Fixed
I-02	Hardcoded `migration_sqrt_price` can leave some tokens unmigrated if config changes	Informational	Acknowledged

Severity Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact

- **High -** leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium -** leads to a moderate material loss of assets in the protocol or moderately harms a group of users. Alternatively, breaking a core aspect of the protocol's intended functionality
- **Low -** leads to a minor material loss of assets in the protocol or harms a small group of users.

Likelihood

- **High -** attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- **Medium -** the attack/vulnerability requires minimal or no preconditions, but there is limited or no incentive to exploit it in practice
- **Low -** requires highly unlikely precondition states, or requires a significant attacker capital with little or no incentive.

Findings

[M-01] `CREATE_POOL_RENT_FEE` is subtracted from the quote token amount unnecessarily

Description

Upon migration, `token_b_amount` (quote token amount added to the LP) is calculated as follows:

```
let token_b_amount = ctx
   .accounts
   .bonding_curve
   .real_sol_reserves
   .checked_sub(ctx.accounts.global.migrate_fee_amount)
   .ok_or(ContractError::ArithmeticError)?
   .checked_sub(CREATE_POOL_RENT_FEE)
   .ok_or(ContractError::ArithmeticError)?;
```

It subtracts the `migrate_fee_amount` and the `CREATE_POOL_RENT_FEE` from the `real_sol_reserves`

After creating the liquidity positions, while the `migrate_fee_amount` is later paid to the `fee_receiver`, the `CREATE_POOL_RENT_FEE` is not transferred anywhere. This means that the `CREATE_POOL_RENT_FEE` (worth `60_000_000` lamports, ~$10) would remain stuck in the `bonding_curve_sol_escrow`.

Recommendation

Consider removing the subtraction of the `CREATE_POOL_RENT_FEE` to ensure that there are no leftover tokens, so that all the required SOL is added to the liquidity positions. Note that if this is done, the migration sqrtPrice would need to be changed accordingly.

Alternatively, consider adding logic to transfer the `CREATE_POOL_RENT_FEE` to the appropriate address to ensure it is not lost.

Remediation: Fixed in commit b2d318b2b9dae4291999f3f2d20c2b4d065f4e66

[M-02] Migration can fail due to rounding differences after splitting liquidity between the platform and dev

Description

The protocol computes the total liquidity to add to Meteora from available tokens using `get_liquidity_for_adding_liquidity`, then splits this `initial_liquidity` between the platform and the dev, based on the set `dev_fee_allo_bps`.

Then the `platform_liquidity` is added during the `create_pool` call, while the `dev_liquidity` is added during a separate call to `add_liquidity`.

When Meteora recalculates the required token amounts for each liquidity portion using `get_amounts_for_modify_liquidity`, it applies `Rounding::Up` independently to each call. This can cause the sum of the individually rounded token amounts (for the platform and dev) to exceed the original token balances used to compute the total `initial_liquidity`, causing the migration to fail due to insufficient token balance.

Note: The issue exists for specific `dev_fee_allo_bps` values, for example as shown in the POC any non-zero `dev_fee_allo_bps` under 5% will cause the issue. The `dev_fee_allo_bps` values that do not cause rounding error failures are shown in `out.txt` after running the POC.

Proof of concept

This bug can be reproduced by changing the `devFeeAlloBps` in `clients-codama/js/src/constants.ts` to any value less than `500`, and then running the bankrun tests- which will fail.

The issue also arises for most values greater than `500`, with a few exceptions.

Recommendation

Consider splitting the available token amounts between the platform and dev first, then independently calculating the corresponding liquidity for each portion.

Here is a sample diff that fixes the issue: https://gist.github.com/ObsidianAudits/5d6af817a85e43ce14ab3f790e3e1f5d

To apply the proposed changes, add the provided diff to the codebase as `diff.txt`, then run `git apply diff.txt`

Remediation: Fixed in commit b2d318b2b9dae4291999f3f2d20c2b4d065f4e66

[I-01] Unused `associated_token_program` account in MigrateDammV2 account struct

Description

The `associated_token_program` account is included in the `MigrateDammV2` account struct, but it is never used.

Consider removing the `associated_token_program` field from the `MigrateDammV2` account struct.

Remediation: Fixed in commit b0dbdf512ad4b5537e5932aadfe8867612564560

[I-02] Hardcoded `migration_sqrt_price` can leave some tokens unmigrated if config changes

Description

The `migration_sqrt_price` is hardcoded instead of being derived from the actual `token_a_amount` and `token_b_amount` calculated during migration. If parameters like `initial_real_token_reserves`, `migration_token_allocation`, or `migrate_fee_amount` change, the true token ratio may shift.

Because liquidity is calculated using the limiting token relative to `sqrt_price`, any mismatch between the global `migration_sqrt_price` and the actual token ratio can result in one token being fully consumed while the other is only partly used, leaving the remainder unmigrated.

Recommendation

Ensure that whenever global bonding curve parameters (which affect the final liquidity ratios) are changed, `migration_sqrt_price` is updated accordingly.