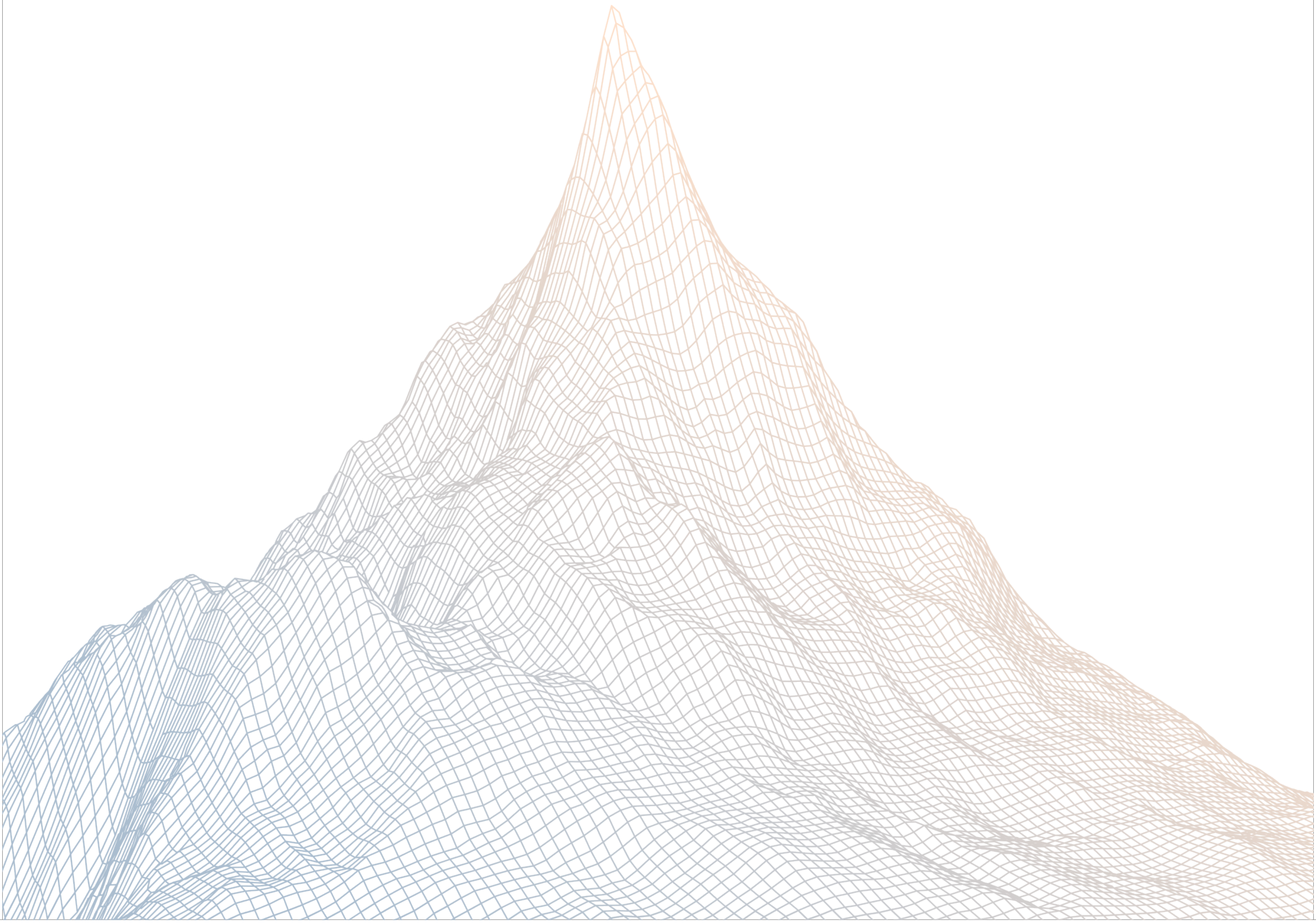
Zenith

Meteora

Smart Contract Security Assessment

VERSION 1.1



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

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	Meteora, DLMM Release 0.11.0
Repository	https://github.com/MeteoraAg/DLMM/pull/443
Commit Hash	`1e8b4a29a46a38b99dfce9ca53d489c4734bcb90`
Files	`Changes in PR-443/r.0.11.0`

Target	Meteora, DLMM Release 0.11.0 Mitigation Review
Repository	https://github.com/MeteoraAg/DLMM/pull/443
Commit Hash	`ae3aeb9a97179cadab11c7bf14ca75b431a27384`
Files	`Changes in PR-443/r.0.11.0`

2.3 Audit Timeline

December 8, 2025	Audit start
December 10, 2025	Audit end
December 17, 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	zap_protocol_fee fails to validate slippage protection	Acknowledged
I-1	migrate_position() should ensure rent is returned to owner or owner specified wallet	Acknowledged

4 Findings

4.1 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] `zap_protocol_fee` fails to validate slippage protection

SEVERITY: Medium	IMPACT: High
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- PR-443

Description:

`validate_zap_out_to_treasury()` in `zap_protocol_fee` will ensure that the next IX is a valid `zap_out` IX. This ensures that the operator cannot pass in wrong parameters to steal the withdrawn protocol fee.

However, it fails to ensure that the slippage value is properly set. This allows the operator to set it to zero and then perform a sandwich attack by manipulating the swap pool before the `zap_protocol_fee` and `zap_out` IX.

The likelihood is low as operator is trusted to enforce a valid slippage value, though it is recommended to mitigate the risk of a sandwich attack.

Recommendations:

This issue can be resolved by ensuring that there are no other IXs before `zap_protocol_fee` and `zap_out`. If necessary, it should be only whitelisted IXs that are not interacting with the swap programs.

Though it is still technically possible to sandwich with malicious swap TXs before and after the zap TX, this should be mitigated by ensuring operator is trusted to pass in a valid slippage value.

Meteora: Acknowledged. In order to do this, operator need to prepare amount in firstly in their wallet, that is not feasible.

Zenith: Acknowledged by client as they are using a trusted operator, and is trusted not to perform a sandwich attack.

4.2 Informational

A total of 1 informational findings were identified.

[I-1] `migrate_position()` should ensure rent is returned to owner or owner specified wallet

`SEVERITY`: Informational	`IMPACT`: Informational
`STATUS`: Acknowledged	`LIKELIHOOD`: Low

Target

- PR-443

Description:

`migration_position()` does not ensure that the rent is refunded to the position's owner when called by the admin.

This is inconsistent with `close_position()`, which will ensure `rent_receiver.key() == position.owner()` when the caller is not the position's owner.

Recommendations:

Consider adding a check to ensure `rent_receiver.key() == position.owner()` when not called by owner.

Meteora: Acknowledged. Admin needs to pay rent fee for PositionV2, that is causing more rent fee than what they collected from PositionV2.