AEGIS_DFM

Security Review

Cantina Managed review by:

Om Parikh, Security Researcher
Jonatas Martins, Associate Security Researcher

September 10, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Solo Labs designs innovative yield solutions that provide consistent returns in the volatile crypto ecosystem.

From Sep 1st to Sep 4th the Cantina team conducted a review of AEGIS_DFM on commit hash e483d9ba. The team identified a total of **8** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	3	2	1
Gas Optimizations	0	0	0
Informational	5	4	1
Total	**8**	**6**	**2**

3 Findings

3.1 Low Risk

3.1.1 Use safeTransferFrom to avoid cases for certain tokens which can lead to loss of funds

Severity: Low Risk

Context: FullRangeLiquidityManager.sol#L1080-L1090

Description: some tokens return `false` instead of reverting, this would use funds reserved for sweeping if `transferFrom` user to `address(this)` fails for some tokens. See:

- no-revert-on-failure.
- missing-return-values.

Recommendation: `FullRangeLiquidityManage` should use `safeTransfer` instead of `transfer`.

Solo Labs: Fix can be found in commit d8e5963c.

Cantina Managed: Fix reviewed.

3.1.2 Round fees against the swapper to avoid potential zero-fee and precision edge cases

Severity: Low Risk

Context: Spot.sol#L215-L216, Spot.sol#L349-L350

Description: In `_beforeSwap` and `_afterSwap` hooks, the fees are rounded in direction of swapper (against LPs & protocol), this allows to pay significantly lower fees or zero fees in case of lower precision tokens (such as GUSD, WBTC). Also, this will pass zero or lowered dynamic fees to `PoolManager` which harms the LPs.

Recommendation: Consider using `mulDivRoundingUp` in `swapFeeAmount` & `hookFeeAmount` and also add / update the test cases, this also aligns with the uniswap's pool behaviour.

Solo Labs: Fixed in 6dee7181.

Cantina Managed: Fix verified.

3.1.3 Hook fees are preemptively charged on full amountSpecified instead of actual used amount

Severity: Low Risk

Context: Spot.sol#L211

Description: In `Spot._beforeSwap`, if `sqrtPriceLimitX96` specificed in swap params is reached during the `PoolManager.swap` then full `amountSpecified` will not be used. However, the hooks fees are currently charged on `params.amountSpecified` which implies whenever swap is limited by `sqrtPriceLimitX96`, users always pays more than required. The differential can be large in certain cases where amount specific is very high and it is expected to always swap upto limit price.

Recommendation:

- Take a cut in `_afterSwap`, but at that time input token would have been used (either partially or fully), so it might require taking fees in output token which is not 1:1 with pool's behaviour.
- Try simultaing a swap in before swap with `StepComputations` and `SwapResult` similar to core's swap function, this might be gas intensive and still would require passing some fee value for carrying out simulation.
- Revert if exact input swap was limited by `sqrtPriceLimitX96`, this may significantly reduce the usability of a hook.

Solo Labs: At this time we will not be altering this logic, we will clarify our documents to make sure it is specified this is how the pool works. Simulating a swap will consume too much gas per swap for it to make that option worth it, and we would like to keep the behavior as close to possible as we can with V3. We will keep an eye on our issue we raised in their repository and see if we could improve on this in a way that keeps existing behavior and doesn't involve simulating a swap.

Cantina Managed: Acknowledged.

3.2 Informational

3.2.1 Consistency with Uniswap V3 oracle when writing observations

Severity: Informational

Context: Spot.sol#L441

Description: In Uniswap V3, the oracle only writes observations when `ModifyLiquidityParams.liquidityDelta != 0`. In Uniswap V4, when adding zero liquidity, the `_beforeRemoveLiquidity` hook is still called, Hooks.sol#L203.

There is no direct impact other than consistency, as there is already a check preventing multiple observations from being added in the same block.

Recommendation: Check for `liquidityDelta != 0` in the `_beforeRemoveLiquidity` function to maintain consistency with Uniswap V3.

Solo Labs: Fix can be found in commit c12282cb9.

Cantina Managed: Fix reviewed.

3.2.2 Code improvements

Severity: Informational

Context: PoolPolicyManager.sol#L378, PoolPolicyManager.sol#L426

Description/Recommendation: This finding is a list of code improvements and the recommendations:

1. Use constants instead of literal numbers for better consistency. Replace the value `10_000_000` with `MAX_SURGE_FEE_MULTIPLIER_PPM` in the revert error within the `PoolPolicyManager.setSurgeFeeMultiplierPpm` function.

2. Simplify the check in the `PoolPolicyManager.setBaseFeeFactor` and `PoolPolicyManager.setPoolDailyBudgetPpm` functions, as the variable will always be `!= 0` in this condition, making the `< 1` check redundant:

```
-   if (factor != 0 && (factor < 1 || factor > MAX_BASE_FEE_FACTOR_PPM)) {
+   if (factor != 0 && factor > MAX_BASE_FEE_FACTOR_PPM) {
      revert Errors.ParameterOutOfRange(factor, 1, MAX_BASE_FEE_FACTOR_PPM);
  }
```

```
-   if (newBudget != 0 && (newBudget < 1 || newBudget > 10 * PrecisionConstants.PPM_SCALE)) {
+   if (newBudget > 10 * PrecisionConstants.PPM_SCALE) {
      revert Errors.ParameterOutOfRange(newBudget, 1, 10 * PrecisionConstants.PPM_SCALE);
  }
```

3. Remove unused variables & imports. Exact locations can be checked via `forge cache clean && forge build` (on latest foundry version).

Solo Labs: Fixes can be found in the commits c12282cb, 9b1acc3e and db037704.

Cantina Managed: Fix reviewed.

3.2.3 `Spot` contract lacks `virtual` keyword for function

Severity: Informational

Context: Spot.sol#L99

Description: The `Spot` contract has been changed to be a base contract and to be inherited, changing most functions back to virtual. However, it still lacks the necessary change to make the `getHookPermission` function compatible with these new changes.

Recommendation: Add the `virtual` modifier to the `getHookPermission` function.

Solo Labs: Fixed in commit 23b90f56.

Cantina Managed: Fix reviewed.

3.2.4 Default max tick per block might never be used

Severity: Informational

Context: PoolPolicyManager.sol#L468

Description: In `PoolPolicyManager`, when `defaultMaxTicks` is 0, the contract uses `DEFAULT_MAX_TICKS_PER_BLOCK` instead. However, after pool initialization, this value is set to at least 1. The check that reverts when `defaultMaxTicks` equals 0 prevents the owner from setting it back to 0 to use the default value. As a result, the `DEFAULT_MAX_TICKS_PER_BLOCK` variable is only used before the pool is initialized.

Recommendation: Consider either allowing `defaultMaxTicks` to be set to 0 or removing both the `DEFAULT_MAX_TICKS_PER_BLOCK` constant and its related `set` functions. Note that this change would affect how the value 0 is handled when the pool is not initialized-it should never be used for non-initialized pools.

Solo Labs: Fixed in the commit 7bc81617

Cantina Managed: Fix reviewed.

3.2.5 A large swap could be broken into smaller chunks to escape capping in `perSwapMode`

Severity: Informational

Context: Spot.sol#L274-L277

Description: one can always break total swap amount into ”n” chunks such that resulting price impact and fees are same as single large swap (except rounding), so one can always prevent capping by swapping size such that movement is `maxTicks - 1`.

Due to this, `updateCapFrequency` will return early. This imposes relatively large risks to illiquid & low volume pools (such as PEPE-SHIBA). If done so, it would have following consequences:

- Cap event won't be triggered even if full large swap is carried out atomically. This benefits all subsequent swappers, including arbitrage and MEV bots who will get better execution.
- It allows decaying and reducing frequency where it ideally shouldn't.

Recommendation: Consider documenting this behavior for pools using `perSwapMode` and risks for liquidity providers.

Solo Labs: Acknowledged.

Cantina Managed: Acknowledged.