Security Review

Cantina Managed

SKY: DPAU V1.15.0

Cantina Managed review by:
Christoph Michel, Lead Security Researcher
M4rio.eth, Lead Security Researcher

CANTINA

October 28, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 Reported deficit can stay 0 while debt remains . . . 4
- 3.2 Informational . . . 4
  - 3.2.1 Hub/Spoke token disagreement can make deposits mint shares for the wrong asset 4
  - 3.2.2 Repeated rounding-based value loss for dust deposits . . . 5
  - 3.2.3 `setMaxSlippage` accepts `maxSlippage >= 1e18` . . . 5

1 Introduction

1.1 About Cantina

Cantina is a security platform that pairs a world-class network of security researchers with agentic AI to help teams building onchain find and fix real risk. Through managed reviews, competitions, and bounties, Cantina secures protocols before vulnerabilities can be exploited.
Learn more at cantina.security

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

From Oct 5th to Oct 6th the Cantina team conducted a review of diamond-pau on commit hash 71604ddc. The team identified a total of **4** issues:

Issues Found

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

The Cantina Managed team reviewed Sky's diamond-pau holistically on commit hash c5c961a8 and concluded that all findings were addressed and no new vulnerabilities were identified.

2.1 Scope

The security review had the following components in scope for diamond-pau on commit hash 71604ddc:

```
src/facets/aave-v4/AaveV4Facet.sol
src/facets/aave-v4/IAaveV4Controller.sol
src/facets/aave-v4/IAaveV4Facet.sol
src/facets/aave/IAaveController.sol
src/facets/basin/IBasinController.sol
src/facets/cctp/ICCTPController.sol
src/facets/centrifuge/ICentrifugeController.sol
src/facets/curve/CurveFacet.sol
src/facets/curve/ICurveController.sol
src/facets/dai-usds/IDAIUSDSController.sol
src/facets/erc4626/IERC4626Controller.sol
src/facets/erc7540/IERC7540Controller.sol
src/facets/ethena/IEthenaController.sol
src/facets/farm/IFarmController.sol
src/facets/layer-zero/ILayerZeroController.sol
src/facets/maple/IMapleController.sol
src/facets/merkl/IMerklController.sol
src/facets/nfat-halo/INFATHaloController.sol
src/facets/nfat-prime/INFATPrimeController.sol
src/facets/otc/IOTCController.sol
src/facets/pendle/IPendleController.sol
src/facets/psm/IPSMController.sol
src/facets/psm3/IPSM3Controller.sol
src/facets/spark-vault/ISparkVaultController.sol
src/facets/superstate/ISuperstateController.sol
src/facets/transfer-asset/ITransferAssetController.sol
src/facets/uniswap-v3/IUniswapV3Controller.sol
src/facets/uniswap-v3/UniswapV3Facet.sol
src/facets/uniswap-v4/IUniswapV4Controller.sol
src/facets/uniswap-v4/UniswapV4Facet.sol
src/facets/usds/IUSDSController.sol
src/facets/weeth/IWEETHController.sol
src/facets/wrap-proxy-eth/IWrapProxyETHController.sol
src/facets/wsteth/IWSTETHController.sol
src/libraries/BeaconConfig.sol
src/libraries/RateLimitHelpers.sol
```

3 Findings

3.1 Low Risk

3.1.1 Reported deficit can stay 0 while debt remains

Severity: Low Risk.

Context: AaveV4Facet.sol#L146-L155.

Description:

`deposit` compares `Hub.getAssetDeficitRay` with `maxDeficits`. Aave reports that deficit after the liquidated collateral is empty and the position has no other active collateral. Dust on a second collateral keeps `activeCollateralCount > 1`, so the reported deficit stays `0` while debt remains.

Trail of Bits logged this as `TOB-AAVE-1` (Medium). Aave marked it Risk Accepted. Their note says the dust exclusion is intentional because of complexity and gas, and that liquidators can keep liquidating until one collateral remains, even when those liquidations are not profitable. Until that happens, a deposit with `maxDeficit == 0` still passes. A compromised allocator can supply rate-limited PAU funds into that shortfall. Each call is limited by its rate-limit bucket. The bucket refills, and more than one Spoke key can point at the same Hub asset. The loss shows up later, when suppliers cannot fully exit.

Recommendation: Accept this and document it. The docs should say a zero `maxDeficit` still allows a deposit while dust collateral is hiding the shortfall.

Sky: Fixed in PR 231.

Cantina Managed: Fix verified.

3.2 Informational

3.2.1 Hub/Spoke token disagreement can make deposits mint shares for the wrong asset

Severity: Informational.

Context: AaveV4Facet.sol#L142-L177.

Description: Note that DPAU treats reserve remaps as a relevant integration risk: the deposit key includes the Spoke-reserve-reported `(hub, assetId, underlying)`, so changing any of them creates an unconfigured key and blocks further deposits. The withdrawal key omits these values so existing positions can still be withdrawn after a remap.

However, this only detects changes reported by the Spoke. The facet does not verify that the Hub associates the same `assetId` with the same underlying token.

If a Hub upgrade changes the underlying associated with an existing asset ID from token A to token B while the Spoke still reports token A, the deposit key remains unchanged and the following can occur:

```
AaveV4Facet.deposit(...)
└─ Spoke.supply(reserveId, tokenAAssetsIn, ALMProxy)
   ├─ tokenA.transferFrom(ALMProxy, Hub, tokenAAssetsIn)
   └─ Hub.add(assetId, tokenAAssetsIn)
      // Hub treats assetId as token B
      // mints shares accounted in token-B units
      // Spoke.getUserSuppliedAssets values the shares in token-B units
```

The Spoke transfers token A directly to the Hub and then calls `Hub.add`.

`Hub.add` does not verify which token was transferred by the current deposit. It only checks whether the Hub's aggregate token-B balance covers the asset ID's recorded liquidity. The deposit can therefore succeed if the Hub already holds enough token B.

As a result, the ALMProxy can spend token A but receive shares accounted and redeemable in token B.

Note that this requires an inconsistent Hub upgrade, the current Hub implementation does not expose a normal asset-remapping function. This check extends DPAU's existing remap protections, but does not protect against malicious Aave governance.

Recommendation:

Before depositing, verify that the Hub and Spoke agree on the reserve's underlying token:

```
(address hubUnderlying,) =
    IAaveV4Hub(reserve.hub).getAssetUnderlyingAndDecimals(reserve.assetId);

require(
    hubUnderlying == reserve.underlying,
    "AaveV4Lib/invalid-hub-asset-metadata"
);
```

This protects both relevant remapping cases for new deposits:

1. Only the Hub changes: The Hub and Spoke report different underlyings, so the new equality check reverts.
2. Both the Hub and Spoke change / Only Spoke changes: The changed spoke underlying produces a new deposit rate-limit hash that is unconfigured and will revert.

Do not apply it to withdrawals, because doing so could lock an existing position after an inconsistent dependency upgrade. The deposit check prevents increasing exposure while the unrestricted withdrawal path preserves the ability to exit.

Sky: Fixed in PR 230.

Cantina Managed: Fix verified.

3.2.2 Repeated rounding-based value loss for dust deposits

Severity: Informational.

Context: AaveV4Facet.sol#L165-L180.

Description:

The slippage check rounds down the minimum acceptable output (RHS), allowing significant relative losses for dust deposits. For example, if the proxy deposited `amount == 2` but received only rounded down shares that are valued at `amountReceived == 1`, then for `maxSlippage < 1e18` the check passes because:

`amountReceived = 1 >= 2 * (1e18 - 1) / 1e18 = 1`

Repeated deposits could therefore leak half of the proxy value to Aave V4. Note that this attack comes with a gas cost and no direct profit to the relayer, so the incentive is unclear.

Recommendation: Accept the risk or round up the minimum-output calculation (RHS). Note that rounding-up would require a reassessment of the configured `maxSlippage` values, as rounding up may cause previously accepted deposits to revert.

Sky: Acknowledged.

Cantina Managed: Acknowledged.

3.2.3 `setMaxSlippage` accepts `maxSlippage >= 1e18`

Severity: Informational.

Context: AaveV4Facet.sol#L110-L121.

Description:

`setMaxSlippage` only checks that `spoke` is not zero. and not that must be `maxSlippage <= 1e18`. Deposit then requires:

`amountReceived >= amount * maxSlippage / 1e18`

At `1e18` that is `amountReceived >= amount`. The parameter comment says `1e18` means no slippage. The comment says `1e18` or above asks for a full credit that Aave's round-down share accounting does not return, so the deposit reverts. The original implementation from grove had this check: ForeignController.sol#L212-L218 .

Recommendation: Add the same check as the original grove implementation.

Sky: Acknowledged.

Cantina Managed: Acknowledged.