CANTINA

Plether

Security Review

Cantina Managed review by:
Red-Swan, Security Researcher

February 13, 2026

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

From Jan 25th to Jan 28th the Cantina team conducted a review of plether-core on commit hash 596b0179. 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	2	2	0
Low Risk	5	4	1
Gas Optimizations	0	0	0
Informational	1	1	0
Total	**8**	**7**	**1**

2.1 Scope

The security review had the following components in scope for plether-core on commit hash 596b0179:

```
src
├── MorphoAdapter.sol
├── SyntheticSplitter.sol
├── SyntheticToken.sol
└── oracles
    └── BasketOracle.sol
```

3 Findings

3.1 Medium Risk

3.1.1 Basket Weights Should Be Constrainted to Unit Value

Severity: Medium Risk

Context: *(No context files were provided by the reviewer)*

Description: `BasketOracle` does not demand that its weights sum to one. In `latestRoundData`, this would cause the `_checkDeviation` check to fail because the price comparison would be in different numéraires or skip that check entirely and return zero prices if all the weights were zero.

Recommendation: Consider adding a `require` statement that enforces the basket weights to sum to one.

Plether: Fixed in commit e7573e5f.

Cantina Managed: Fix verified.

3.1.2 Adapter Asset Doesn't Account For Accrued Interest

Severity: Medium Risk

Context: *(No context files were provided by the reviewer)*

Description: `MorphoAdapter.totalAssets` has comments that state it returns accrued interest but the Morpho Blue documentation (under *Best Practices*) states:

> Interest Accrual: Remember that `totalBorrowAssets` and `totalSupplyAssets` are only updated when an interaction triggers `_accrueInterest`.

Morpho's `position` method therefore can't call `_accrueInterest` because it is a `view` function and indeed upon inspecting `Morpho.sol` we see that interest is only accrued when funds enter/exit the contract.

This feeds into many other ERC-4626 methods like `maxWithdraw`, `previewDeposit`, and `convertToShares` as well as into `SyntheticSplitter` such as `harvestYield` and `previewBurn`.

Recommendation: Consider accruing interest in `MorphoAdapter.totalAssets` or utilizing `MorphoBalancesLib`'s `expectedTotalSupplyAssets` if you wish to save gas.

Plether: Fixed in commit fc7a305d.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 Duplicate Event Emissions

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: The following functions do not check that the new item being written to storage is not the same as the old and therefore will fire an event when nothing noteworthy has changed. Doing so may mislead off-chain systems.

- `BasketOracle.proposeCurvePool`.
- `MorphoAdapter.setUrd`.
- `SyntheticSplitter.proposeAdapter`.

Recommendation: Consider reverting if the item to be written is the same as what is already in storage.

Plether: Acknowledged.

Cantina Managed: Acknowledged.

3.2.2 Missing Security Contact In Source Files

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: The smart contract files do not include a security contact (such as an email address, a link to a bug bounty program, or a dedicated security policy) in the contracts that will be deployed. It is often the case that third-parties will only interact with the code through other platforms like sourcify, blockscout, or etherscan rather than the code's actual repository. In the event that a vulnerability is discovered by a third-party researcher or a ”white-hat” hacker a contact in the source code allows them to easily, reliably, and privately report findings to the development team. Without a clear disclosure process, researchers may resort to public communication (e.g., social media) to get the team’s attention, inadvertently alerting malicious actors to the flaw before it can be patched or mitigated.

Recommendation: Consider adding a security contact at the top of each source file.

Plether: Fixed in commit 819111c3.

Cantina Managed: Fix verified.

3.2.3 `AggegatorV3Interface` Conformity

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: `BasketOracle`'s `getRoundData` function doesn't strictly conform to the `AggregatorV3Interface` in that it doesn't actually return data for the input round ID.

Recommendation: Consider reverting when this method is called, or alternatively, revert only when called with a round ID that is not the latest round ID. Either approach would provide clear error messaging and prevent silent data corruption in external integrations that expect standard Chainlink oracle behavior.

Plether: Fixed in commit 66ed9863.

Cantina Managed: Fix verified.

3.2.4 Owner Can Circumvent Rescue Constraint

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: In SytheticSplitter.rescueToken, the owner is not allowed to transfer USDC out of the contract. However, they are still able to transfer yield adapter tokens out, which are redeemable for 90% of the contract's USDC.

Recommendation: Consider reverting if `token == address(yieldAdapter)`.

Plether: Fixed in commit 017c09c7.

Cantina Managed: Fix verified.

3.2.5 New Adapter Could Have Different Underlying Asset

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: In `proposeAdapter` and `finalizeAdapter`, it is possible, however unlikely, that the new adapter have a different underlying asset than the previous adapter or `SyntheticSplitter`'s `USDC`. If this is the case it would throw all math and assumptions of the contract up in the air.

Recommendation: Consider requiring that `pendingAdapter.asset() == USDC` in `proposeAdapter` to ensure that the new adapter shares the same asset.

Plether: Fixed in commit 36598c36.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 Code Accuracy

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description and Recommendations: There are several locations where the code can be made to align better with the intent of the protocol.

- `SyntheticSplitter`'s `TOKEN_A` and `TOKEN_B` have specific roles within the system and are referred to as BEAR and BULL constantly. Consider naming them as such.
- `MorphoAdapter`'s `marketParams` is set in storage only once and there can be immutable.
- `SyntheticSplitter`'s `MIN_SURPLUS_THRESHOLD` is hard-coded to `USDC`'s 6 decimals while `USDC_MULTIPLIER` is not. Consider whether you should hard code everthing to six decimals or not.
- `BasketOracle`'s `getRoundData` creates an external call to what could be an internal function. Consider making `latestRoundData` `public` and making this call internal.
- `CAP` is defined but not used by `BasketOracle`. Consider removing it.
- In `previewHarvest` the variable `adapterAssets` represents the same concept as `harvestYield`'s `totalAssets` but they have different names. Consider changing `harvestYield`'s variable to match `previewHarvest`'s.

Plether: Fixed in commit 8e72d096.

Cantina Managed: Fix verified.