CANTINA

Plutus PR 2

Security Review

Cantina Managed review by:

R0bert, Lead Security Researcher

Jonatas Martins, Associate Security Researcher

October 1, 2025

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**

3 Findings **4**
3.1 Medium Risk . . . 4
3.1.1 Rescue helper can be griefed via public relock to permanently stall balances . . . 4
3.2 Low Risk . . . 4
3.2.1 Rewards are not rescued in the `rescueExpiredLocksForPlutus` function . . . 4
3.3 Informational . . . 4
3.3.1 Missing NatSpec in `rescueExpiredLocksForPlutus` function . . . 4

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

PlutusDAO is an Arbitrum-native governance aggregator that maximizes user liquidity and rewards by aggregating the governance power of various decentralized finance (DeFi) protocols through its governance token, PLS.

From Sep 22nd to Sep 23rd the Cantina team conducted a review of radiant-core-shared on commit hash 8a013dfc. The team identified a total of **3** issues:

Issues Found

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

3 Findings

3.1 Medium Risk

3.1.1 Rescue helper can be griefed via public relock to permanently stall balances

Severity: Medium Risk

Context: MultiFeeDistribution.sol#L678-L693, MultiFeeDistribution.sol#L695-L715

Description: `rescueExpiredLocksForPlutus` was added so the owner can sweep matured locks from 0x2A2CAFbB...5197 into the Plutus multisig, but it still assumes `_cleanWithdrawableLocks` will find expired positions and simply transfers whatever amount it receives. A third party can always call the public helper `withdrawExpiredLocksForWithOptions(address_, limit_, false)` on that same wallet right before the rescue executes. Inside `_withdrawExpiredLocksFor` the branch:

```
if (isRelockAction || (address_ != msg.sender && !autoRelockDisabled[address_])) {
    _stake(amount, address_, defaultLockIndex[address_], true);
}
```

restakes the full matured balance because `autoRelockDisabled[address_]` is `false` for the Plutus wallet. Once that happens `_cleanWithdrawableLocks` returns zero to the rescue path, so the ERC20 transfer moves nothing and the new tranche is locked for another 360 days. The Foundry fork test shared: `test/MultiFeeDistributionRescueFrontRun.t.sol` shows this concretely: scenario 1 sweeps ~1.58 M tokens when the rescue runs first, scenario 2 shows a third party re-locking the same amount so the rescue emits no transfer and leaves a fresh one-year lock in place. The impact is a persistent denial-of-service on the owner's administrative recovery: as long as anyone keeps re-locking before each attempt, the ~1.58 M tokens can never be migrated.

Proof of Concept: See gist `https://gist.github.com/r0bert-ethack/aaa81613be4f661ae130fdcf45f3db6b`.

Recommendation: Extend the rescue flow to neutralize auto re-lock before attempting the sweep. One option is to have the owner call `setRelock(false)` on the target wallet (e.g., expose an owner-only override that sets `autoRelockDisabled[target] = true`) and only then invoke `_cleanWithdrawableLocks`, so the relock branch can never trigger.

Plutus DAO: Acknowledged.

Cantina Managed: Acknowledged.

3.2 Low Risk

3.2.1 Rewards are not rescued in the `rescueExpiredLocksForPlutus` function

Severity: Low Risk

Context: MultiFeeDistribution.sol#L701

Description: While the `_updateReward` function updates rewards, it doesn't claim them. This function is designed to rescue funds from locks in the `address_` to `plutusMS`, but it doesn't transfer the earned rewards to the `plutusMS` address as well. As a result, these rewards might be stuck in the `address_`.

Recommendation: Consider implementing a reward distribution from `address_` to `plutusMS`. A similar approach to the `claimFromConverter` function could be used.

Plutus DAO: Acknowledged.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Missing NatSpec in `rescueExpiredLocksForPlutus` function

Severity: Informational

Context: MultiFeeDistribution.sol#L695

Description: The `rescueExpiredLocksForPlutus` function is missing the NatSpec comments. These comments are important for better code comprehension and documentation.

Recommendation: Add NatSpec comments to the `rescueExpiredLocksForPlutus` function.

Plutus DAO: Acknowledged.

Cantina Managed: Acknowledged.