Silo Labs v2

Security Review

Cantina Managed review by:
Rikard Hjort, Lead Security Researcher
Kankodu, Security Researcher

February 19, 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

The Silo Protocol is a non-custodial lending primitive that creates programmable risk-isolated markets known as silos.

From Jan 23rd to Jan 28th the Cantina team conducted a review of silo-contracts-v2 on commit hash f77309dd. The team identified a total of **2** 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	1	0	1
Total	**2**	**1**	**1**

2.1 Scope

The security review had the following components in scope for silo-contracts-v2 on commit hash f77309dd:

```
silo-core/contracts
├── Silo.sol
├── SiloConfig.sol
├── hooks
│   ├── SiloHookV2.sol
│   ├── defaulting
│   │   ├── DefaultingRepayLib.sol
│   │   ├── DefaultingSiloLogic.sol
│   │   └── PartialLiquidationByDefaulting.sol
│   └── liquidation
│       └── PartialLiquidation.sol
└── incentives
    ├── SiloIncentivesController.sol
    └── SiloIncentivesControllerFactory.sol
```

3 Findings

3.1 Low Risk

3.1.1 Rounding Up could be Unsafe in Minimal Collateral Scenarios

Severity: Low Risk

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

Description: In earlier version of the Silo, an error existed where borrowing a large amount against just 1 wei of protected collateral prevented liquidation. During liquidation, the system calculates the amount of collateral assets to transfer to the liquidator. However, with only 1 wei of collateral, the calculation in the following function rounds down, resulting in zero collateral assets being allocated to the liquidator:

```
/// @notice reverts on `_totalValue` == 0
/// @dev calculate assets based on ratio: assets = (value, totalAssets, totalValue)
/// to calculate assets => value, use it like: value = (assets, totalValue, totalAssets)
function valueToAssetsByRatio(uint256 _value, uint256 _totalAssets, uint256 _totalValue)
    internal
    pure
    returns (uint256 assets)
{
    require(_totalValue != 0, IPartialLiquidation.UnknownRatio());

    // round UP to allow for 1 wei collateral liquidations and
    // always take more collateral and debt during liquidation
    assets = _value * _totalAssets / _totalValue;
}
```

This causes the liquidation to fail with a `NoRepayAssets` error due to this the check in PartialLiquidationExecLib.sol#L164.

To address this, a rounding-up mechanism was introduced:

```
assets = Math.mulDiv(_value, _totalAssets, _totalValue, Rounding.UP);
```

This prevents the zero-allocation issue and allows liquidation to proceed.

However, this fix introduces a new problem. If a liquidator repays even a small fraction of the debt (relative to the collateral's value), they can claim the entire 1 wei of collateral. This enables bad actors to intentionally borrow large amounts against 1 wei of collateral and then self-liquidate, creating bad debt within the protocol.

Recommendation: To mitigate this attack vector, ensure that 1 wei of collateral does not represent a significant dollar value in the first place, rather than relying on the rounding fix alone. This can involve setting minimum collateral requirements.

Silo Finance: Fixed in PR 1701.

Cantina Managed: Akthough we could not make a working proof of concept, theoretically rounding up could be weaponised against the protocol in the unlikely scenarios of using 1 wei as collateral.

Silo Finance: The benefits from rounding up (liquidation being possible) are not worth the risk of potentially introducing bad debt in this unlikely scenario. We'll keep the rounding direction as down but retained the use of `mulDiv`. See PR 1753.

3.2 Informational

3.2.1 Build issues

Severity: Informational

Context: foundry.toml#L3, README.md#L16

Description: The project does not build according to instructions using the latest stable Foundry version 1.5.1 on some machines, in particular on certain macOS machines. Deployment for the local test setup

fails with `[FAIL: InvalidUopt()] setUp() (gas: 0)` for large parts of the test suite. Version 1.2.3 is used on CI and can be used reliably to build on most machines and has been recommended by developers.

Recommendation: Document in README.md build instructions that Foundry v1.2.3 is the supported build version. Consider updating support to v1.5.1.

Silo Finance: Acknowledged.

Cantina Managed: Acknowledged.