4th05

Silo Finance

Security Review Report

31 March 2025

Security Review Report

4th05

March 31, 2025

Table of Contents

Protocol Summary

The Silo Protocol is a non-custodial lending primitive that creates programable risk-isolated markets known as silos. Any user with a wallet can lend or borrow in a silo in a non-custodial manner. Silo markets use the peer-to-pool, overcollateralized model where the value of a borrower's collateral always exceeds the value of their loan.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Overview

Contest platform	Code4rena
LOC	1697
Language	Solidity
Commit	0409be5b85d7aabfbbe10de1de1890d4b862d2d5
Previous audits	Cantina, Certora, Sigma Prime, 4naly3er

Scope

- /silo-vaults/contracts/IdleVault.sol
- /silo-vaults/contracts/IdleVaultsFactory.sol
- /silo-vaults/contracts/PublicAllocator.sol
- /silo-vaults/contracts/SiloVault.sol
- /silo-vaults/contracts/SiloVaultsFactory.sol
- /silo-vaults/contracts/incentives/VaultIncentivesModule.sol
- /silo-vaults/contracts/incentives/claiming-logics/SiloIncentivesControllerCL.sol
- /silo-vaults/contracts/incentives/claiming-logics/SiloIncentivesControllerCLFactory.sol
- /silo-vaults/contracts/interfaces/IIncentivesClaimingLogic.sol
- /silo-vaults/contracts/interfaces/INotificationReceiver.sol
- /silo-vaults/contracts/interfaces/IPublicAllocator.sol
- /silo-vaults/contracts/interfaces/ISiloIncentivesControllerCLFactory.sol
- /silo-vaults/contracts/interfaces/ISiloVault.sol
- /silo-vaults/contracts/interfaces/ISiloVaultsFactory.sol
- /silo-vaults/contracts/interfaces/IVaultIncentivesModule.sol
- /silo-vaults/contracts/libraries/ConstantsLib.sol
- /silo-vaults/contracts/libraries/ErrorsLib.sol
- /silo-vaults/contracts/libraries/EventsLib.sol
- /silo-vaults/contracts/libraries/PendingLib.sol
- /silo-vaults/contracts/libraries/SiloVaultActionsLib.sol
- /silo-core/contracts/incentives/SiloIncentivesController.sol
- /silo-core/contracts/incentives/SiloIncentivesControllerFactory.sol
- /silo-core/contracts/incentives/SiloIncentivesControllerGaugeLike.sol
- /silo-core/contracts/incentives/SiloIncentivesControllerGaugeLikeFactory.sol
- /silo-core/contracts/incentives/base/BaseIncentivesController.sol
- /silo-core/contracts/incentives/base/DistributionManager.sol
- /silo-core/contracts/incentives/interfaces/IDistributionManager.sol
- /silo-core/contracts/incentives/interfaces/ISiloIncentivesController.sol
- /silo-core/contracts/incentives/interfaces/ISiloIncentivesControllerFactory.sol
- /silo-core/contracts/incentives/interfaces/ISiloIncentivesControllerGaugeLikeFactory.sol
- /silo-core/contracts/incentives/lib/DistributionTypes.sol

Issues found

Severity	Number of issues found
High	0
Medium	0
Low	1
Info	0

Findings

[L1] Wrong DECIMALS_OFFSET for assets with <6 decimals

Relevant GitHub Links

- https://github.com/code-423n4/2025-03-silo-finance/blob/main/silo-vaults/contracts/SiloVault.sol#L122

Finding description and impact

In the constructor of `SiloVault` contract, `DECIMALS_OFFSET` is not correctly set for assets having less than 6 decimals. For these assets the `DECIMALS_OFFSET` value would be >18, that is not intended by design of the contract, as written in the following code comments:

```
   /// @notice OpenZeppelin decimals offset used by the ERC4626
      implementation.
    /// @dev Calculated to be max(0, 18 - underlyingDecimals) at
       construction, so the initial conversion rate maximizes
    /// precision between shares and assets.
    uint8 public immutable DECIMALS_OFFSET;
```

However, looking at the docs provided for this contest all ERC20 tokens should be able to be used without incurring any possible disruption.

> ERC20 used by the protocol –> Any
>
> Low decimals (< 6) In scope

This wrong value impacts all conversion functions which return a different value from what they should.

```
// Gemini token (with only 2 decimals)
DECIMALS_OFFSET = uint8(UtilsLib.zeroFloorSub(18 + 6, 2)); // --> 22
```

Recommended mitigation steps

Change the code as follows:

```
- DECIMALS_OFFSET = uint8(UtilsLib.zeroFloorSub(18 + 6, decimals));
+ DECIMALS_OFFSET = uint8(UtilsLib.zeroFloorSub(18, decimals));
```