CANTINA

Infinifi Pendle SY

Security Review

Cantina Managed review by:

R0bert, Lead Security Researcher
Slowfi, Security Researcher

July 17, 2025

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	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

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 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

infiniFi is a self-coordinated depositor-driven system designed to tackle the challenges of duration gaps in traditional banking.

From Jul 11th to Jul 14th the Cantina team conducted a review of pendle-sy-public-siusd on commit hash cdd9492b. The team identified a total of **3** issues:

Issues Found

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

3 Findings

3.1 High Risk

3.1.1 Loss of funds due to broken redeem functionality for siUSD

Severity: High Risk

Context: PendleInfiniFisiUSD.sol#L60-L80

Description: In the `PendleInfiniFisiUSD` contract, the `_redeem` function handles the redemption of SY shares back to underlying tokens, supporting USDC, iUSD and siUSD as output tokens based on the `isValidTokenOut` and `getTokensOut` functions, which explicitly include SIUSD (the staked token address) as a valid option. However, when `tokenOut` is set to SIUSD, the function does not properly process the redemption and instead falls through to an uninitialized return statement: `return amountTokenOut;`, where `amountTokenOut` defaults to 0 due to lack of assignment in this path. This occurs because the `if` conditions only handle iUSD and USDC explicitly, leaving siUSD unhandled despite being advertised as supported.The base contract `PendleERC4626UpgSYV2` correctly handles yieldToken (siUSD) redemptions by assigning `amountTokenOut = amountSharesToRedeem` and transferring the siUSD tokens, but the override in `PendleInfiniFisiUSD` neglects this case. As a result, during redemption via `SYBaseUpg.redeem`, the user's SY shares are burned successfully, but no tokens are transferred to the receiver, leading to permanent loss of the redeemed value. Here is the relevant code snippet from `PendleInfiniFisiUSD._redeem`:

```
if (tokenOut == IUSD) {
    return IInfiniFiGateway(GATEWAY).unstake(receiver, amountSharesToRedeem);
} if (tokenOut == USDC) {
    uint256 receiptOut = IInfiniFiGateway(GATEWAY).unstake(address(this), amountSharesToRedeem);
    address redeemController = IInfiniFiGateway(GATEWAY).getAddress("redeemController");
    uint256 assetsOut = IRedeemController(redeemController).receiptToAsset(receiptOut);
    return IInfiniFiGateway(GATEWAY).redeem(receiver, receiptOut, assetsOut);
} return amountTokenOut;
```

In contrast, the `_previewRedeem` function correctly previews the siUSD redemption by returning `amountSharesToRedeem`, creating a discrepancy where users or integrators expect to receive siUSD based on previews but receive nothing in reality. The overall impact is severe, as this bug enables direct and irrecoverable loss of user funds during redemptions targeting siUSD.

Recommendation: To correct this issue, add explicit handling for the SIUSD case in the `_redeem` function to mirror the base contract's behavior, ensuring the `amountSharesToRedeem` is assigned to `amountTokenOut` and the tokens are transferred to the receiver. Alternatively, if siUSD redemption is not intended, remove SIUSD from `getTokensOut` and `isValidTokenOut` to prevent misleading users.Here is a suggested code fix for adding the handling:

```
if (tokenOut == SIUSD) {
    amountTokenOut = amountSharesToRedeem;
    _transferOut(SIUSD, receiver, amountTokenOut); // Assuming _transferOut from TokenHelper is available
    return amountTokenOut;
}
```

infiniFi: Fixed in commit 1cfc3b96.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 PreviewDeposit and `previewRedeem` reports inaccurate rates

Severity: Medium Risk

Context: PendleInfiniFisiUSD.sol#L86-L127

Description: ERC-4626 expressly requires the preview helpers to provide a quote that is "*as close to and no more than the exact result of the real deposit or redeem executed in the same transaction*". The Pendle InfiniFi SY breaks this guarantee whenever the underlying gateway triggers its internal reward-settlement routine.

Both `stake` and `mintAndStake` inside `InfiniFiGatewayV2` begin by calling `yieldSharing.distributeInterpolationRewards()`. That function realizes pending yield for every SIUSD holder and, crucially, mutates the vault's accounting variables before the SY's own balance-changing code executes. In contrast, `previewDeposit` and `previewRedeem` are pure view calls: they bypass the gateway and read the pre-distribution state. The up-to-date exchange-rate that will exist after the first real deposit or redeem in a block is therefore invisible to the previews.

The observable consequence is that an integrator calling `previewDeposit(token, amount)` receives a quote that can be off by the full size of the reward distribution. Every time rewards accrue between two user interactions, the first state-changing call will settle them and invalidate all cached previews observed since the previous settlement, violating ERC-4626 section `previewDeposit` / `previewRedeem`.

Recommendation: Consider updating the `_previewDeposit` and `_previewRedeem` functions in `PendleInfiniFisiUSD` to consider the effects of the `distributeInterpolationRewards` call.

infiniFi: Fixed in commit 1cfc3b96

Cantina Managed: Fix verified. The fix updated the functions `_previewDeposit` and `_previewRedeem` to provide an exact quote, adjusting to `ERC4626` specification.

3.3 Informational

3.3.1 Exchange rate is given directly in iUSD

Severity: Informational

Context: PendleInfiniFisiUSD.sol#L82-L84

Description: The `exchangeRate` function in `PendleInfiniFisiUSD` returns `IERC4626(SIUSD).convertToAssets(PMath.O` which computes the iUSD value per siUSD share. However, while this rate is correct, if iUSD is ever slashed it will not be reflected for any integrators. Integrators might mistakenly interpret the exchange rate as reflecting USDC-equivalent value, especially since deposits and redeems often involve USDC paths.

Without adjustment, integrators querying `exchangeRate` may overvalue positions, leading to incorrect collateralization, liquidation thresholds, or trading decisions in protocols built on Pendle.

Recommendation: Update the `exchangeRate` function to compute the USDC-equivalent value by chaining the iUSD-to-siUSD rate with the iUSD-to-USDC redemption rate from `RedeemController`. This provides a more accurate and integrator-friendly rate. Suggested code fix:

```
address redeemController = IInfiniFiGateway(GATEWAY).getAddress("redeemController");
uint256 iusdPerSiUsd = IERC4626(yieldToken).convertToAssets(PMath.ONE);
return IRedeemController(redeemController).receiptToAsset(iusdPerSiUsd);
```

Additionally, enhance documentation in `assetInfo` and `exchangeRate` to clarify that the rate is slashing-sensitive and recommend using the full USDC conversion for valuations.

infiniFi: Fixed in commit 1cfc3b96 as suggested.

Cantina Managed: Fix verified.