Steakhouse Oracles

Security Review

Cantina Managed review by:

Eric Wang, Lead Security Researcher
Om Parikh, Security Researcher

April 1, 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

Steakhouse builds transparent, efficient, and accessible financial primitives to power the next generation of capital markets on public blockchains.

From Mar 20th to Mar 21st the Cantina team conducted a review of steakhouse-oracles on commit hash 5f248a3e. 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	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	2	0	2
Total	**3**	**1**	**2**

3 Findings

3.1 Low Risk

3.1.1 Unsafe type casting from `uint256` to `int256`

Severity: Low Risk

Context: ERC4626Feed.sol#L98-L100

Description: The `price` variable, returned from the `getPrice()` function, is cast to `int256` to comply with the interface of Chainlink's `latestRoundData()` function.

However, if `price` is large enough, i.e., `price >= (1 << 255)`, casting it to `int256` will result in a negative value. At the time of writing, Morpho checks if the returned `answer` is non-negative (Chainlink-DataFeedLib.sol#L24), so incorrectly cast price would be rejected by Morpho automatically.

Recommendation: Although the case where the price is such an extreme value should be relatively rare, since it likely indicates an error and cannot be represented in `int256`, it would be reasonable to revert with an error instead. For example, using the `SafeCast` library would check if the value fits into `int256`. Additionally, the type casting check could prevent unintended outcomes if other on-chain integrators do not handle the returned negative price.

Steakhouse: Fixed in commit 39b5ad5.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Improving test coverage

Severity: Informational

Context: ERC4626Feed.sol#L66-L69

Description: The `if` branch at L66-L69 is not covered by any test. Consider adding a unit test to cover it or modifying the `testDifferentDecimals()` function in `ERC4626FeedTest.t.sol` to automatically test the creation of the feed with different custom decimals. The following example assumes a maximum decimals of 48, which can be adjusted:

```
function testFuzzDifferentDecimals(uint8 customDecimals) public {
    vm.assume(customDecimals > 0 && customDecimals <= 48);
    ERC4626Feed customFeed = new ERC4626Feed(vault, customDecimals);

    uint256 amount = 2e18; // 2 assets per share
    uint256 shares = 1e18;

    // Mock vault conversion rate - 1 share = 2 assets
    vault.setConvertToAssets(shares, amount);

    // Price should be (2 * 10^customDecimals) since we're converting from 18 decimals to customDecimals
    uint256 expectedPrice = (amount * 10**customDecimals) / 10**DECIMALS;
    assertEq(customFeed.getPrice(), expectedPrice);
}
```

Recommendation: Consider modifying the tests to improve code coverage.

Steakhouse: Acknowledged.

Cantina Managed: Acknowledged.

3.2.2 Security considerations of the oracle design

Severity: Informational

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

Description: The `ERC4626Feed` contract is designed as an oracle that returns the exchange rate between the vault token and the asset token of an ERC-4626 vault through the `convertToAssets()` function. This

design inherently exposes the oracle to the risks of exchange rate manipulation on the vault. To ensure secure integration, the ERC-4626 vault should satisfy the following requirements:

1. The vault and/or the underlying token should implement security measures to prevent exchange rate manipulation attacks, including inflation attacks through direct donation or stealth donation, or a vault reset attack that decreases the exchange rate.
2. If the vault allows stealth donations (e.g., through rounding errors), it should be ensured that stealth donation attacks are inefficient or impractical to execute, e.g., when the vault's total supply and total assets are large enough. This may require continuous monitoring of the vault's on-chain state.
3. Custom functionality of the vault should not introduce additional risks of exchange rate manipulation. For example, the vault should not implement a flash loan functionality that could temporarily decrease the total assets in the vault, therefore affecting the exchange rate.
4. If the vault is upgradeable, it should be monitored for new upgrades, and the changes should be reviewed.
5. The underlying token of the vault should have no vulnerabilities leading to arbitrary manipulation of token balances. Otherwise, it could be exploited to affect the token balance of the vault.
6. The vault or the underlying token may have privileged roles and functions. Privileged roles should be granted and managed securely, as the compromise of the privileged roles could lead to attacks. For example, if the minter/burner role of the underlying token is compromised, it could mint or burn an arbitrary number of tokens to the vault and, therefore, affect the vault's exchange rate.

According to the protocol team, the vault integrated with the `ERC4626Feed` oracle is the wUSDL token, which is a proxy contract with the implementation at address `0x2954C85E7e2B841d0e9A9fdcC09Dac1274057D71`. The vault uses OpenZeppelin v4.9, with minor code changes. Since the underlying token, USDL, disallows direct transfer to the vault, and the on-chain condition of the vault does not allow an efficient stealth donation attack at the time of writing, no immediate risk of exchange rate inflation has been identified.

However, since the wUSDL token and USDL are upgradable, a privileged role may add additional features or changes to the tokens, which should be thoroughly reviewed. The wUSDL and USDL tokens are considered out of the scope of this review.

Recommendation: If the oracle will integrate with other ERC-4626 vaults in the future, verify that the vaults satisfy the above requirements.

Steakhouse: Acknowledged.

Cantina Managed: Acknowledged.