Steakhouse Oracles

Security Review

Cantina Managed review by:

Optimum, Lead Security Researcher
High Byte, Security Researcher

June 23, 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 May 15th to May 17th the Cantina team conducted a review of steakhouse-oracles on commit hash b14f2c68. 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	1	0
Low Risk	0	0	0
Gas Optimizations	0	0	0
Informational	2	1	1
Total	**3**	**2**	**1**

3 Findings

3.1 Medium Risk

3.1.1 Deviation may go undetected due to biased division base `currentBackupPrice`

Severity: Medium Risk

Context: MetaOracleDeviationTimelock.sol#L133

Description: In both `initialize()` and `getDeviation()`, `currentBackupPrice` is used as the denominator as we can see in the following code snippets:

```
// initialize()
initialDeviation = (diff * 10**18) / initialBackupPrice;
```

```
// getDeviation()
return (diff * 10**18) / currentBackupPrice;
```

The implicit assumption behind this code is that the expected deviation is always a drop in the price of backup oracle. While it might be the case in most deployments, it is still possible that in certain scenarios the price of the backup oracle will be greater than the price of the primary oracle. In this case, the deviation might not cross the predefined threshold although it should. To better understand, let's consider the following scenario:

1. The primary oracle tracks the price of BTC/USD, assuming it is $100,000.
2. The backup oracle tracks the price of cbBTC/USDC, assuming it is 100,000 USDC.
3. The deviation threshold is 5%.

If at some point USDC depegs while cbBTC does not, we expect cbBTC/USDC to increase, assuming it is 105,000 USDC at the moment, this will not be considered deviated since `getDeviation()` will return `(105,000 - 100,000) / 105,000 ~ 4.76% < 5%`.

Recommendation: Consider changing the code to divide by the minimum value of the two price instead.

Steakhouse: Fixed in commit 36385342.

Cantina Managed: The issue was fixed by taking the average between the two prices (primary and backup) instead.

3.2 Informational

3.2.1 `initialize()` implements logic that's duplicated in `getDeviation()`

Severity: Informational

Context: MetaOracleDeviationTimelock.sol#L50

Description: The following logic in `initialize()` is duplicated from `getDeviation()`:

```
uint256 initialPrimaryPrice = _primaryOracle.price();
uint256 initialBackupPrice = _backupOracle.price();
uint256 initialDeviation;
if (initialBackupPrice == 0) {
    initialDeviation = initialPrimaryPrice == 0 ? 0 : type(uint256).max;
} else {
    uint256 diff;
    if (initialPrimaryPrice >= initialBackupPrice) {
        diff = initialPrimaryPrice - initialBackupPrice;
    } else {
        diff = initialBackupPrice - initialPrimaryPrice;
    }
    initialDeviation = (diff * 10**18) / initialBackupPrice;
}
```

Code duplication is considered bad practice as it increases maintenance effort, reduces readability, and expands the audit surface. It also risks inconsistencies when updating shared logic.

Recommendation: Refactor the shared logic into an internal view function (e.g., `_getDeviation()`) and call it from both `initialize()` and `getDeviation()` to improve maintainability while preserving gas efficiency.

Steakhouse: Fixed in commit 36385342.

Cantina Managed: Fix verified.

3.2.2 The current mechanism will not detect the rare scenario of "double-depegging"

Severity: Informational

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

Description: As communicated with the team, the intended use of this system is to use the primary oracle to track the price of an asset against USD (BTC/USD for example), and the backup oracle to track the price of a wrapped version of that asset with a wrapped version of USD (cbBTC/USDC for example).

In the unlikely scenario of a depegging of both cbBTC and USDC, the backup oracle price might not deviate from the primary oracle price while using the current `getDeviation()` function. Although this scenario is unlikely, in case both assets tracked by the backup oracle are backed by the same centralized entity it might be more likely than we think.

Recommendation: Consider adding a third oracle that will track the price of USDC/USD.

Steakhouse: Acknowledged.

Cantina Managed: Issue was acknowledged without a fix since the described scenario does not violate the expected behavior of the system as in case of double depegging the ratio will be maintained.