CANTINA

Uniswap: Continuous Clearing Auction PR 252

Security Review

Cantina Managed review by:
Kaden, Lead Security Researcher
Phaze, Lead Security Researcher

November 17, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3
- 2.2 Cantina Managed Team Statement . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 `maxBidPrice` derivation step leads to partial overestimation . . . 4
- 3.2 Informational . . . 4
  - 3.2.1 Outdated comment for `MAX_BID_PRICE` calculation . . . 4
  - 3.2.2 Incorrect step documented in `maxBidPrice` derivation . . . 5

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

Uniswap is an open source decentralized exchange that facilitates automated transactions between ERC20 token tokens on various EVM-based chains through the use of liquidity pools and automatic market makers (AMM).

From Nov 11th to Nov 13th the Cantina team conducted a review of continuous-clearing-auction on commit hash 154fd189. 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**

2.1 Scope

The security review had the following components in scope for continuous-clearing-auction on commit hash 154fd189:

```
src
├── ContinuousClearingAuction.sol
├── libraries
│   ├── ConstantsLib.sol
│   └── MaxBidPriceLib.sol
├── TickStorage.sol
└── TokenCurrencyStorage.sol
```

2.2 Cantina Managed Team Statement

Kaden's participation in this review was limited to a single day, specifically November 11th.

3 Findings

3.1 Low Risk

3.1.1 `maxBidPrice` derivation step leads to partial overestimation

Severity: Low Risk

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

Description: In `MaxBidPriceLib.maxBidPrice`, we include the following step in the derivation of the `maxBidPrice`:

```
*  2^(x+y) < L_max * |2^((x+96)/2)-p_sqrtMin|
* Knowing that for large values of x, |2^((x+96)/2)-p_sqrtMin| ~ 2^((x+96)/2), we can
 ↪  simplify to:
*  2^(x+y) < L_max * 2^((x+96)/2)
```

Since we are removing the subtraction of `p_sqrtMin` here while computing a maximum value, we are effectively increasing the maximum value. In practice, this doesn't appear to be impactful as the value used for `L_max` is already an underestimate for the actual required maximum, however, we should aim to always round the maximum value down.

Recommendation: Consider either including `p_sqrtMin` in the derivation, or replacing it with a larger value that is easier to work with such that we further underestimate the `maxBidPrice`.

Uniswap Labs: Fixed in PR 282.

Cantina Managed: Fix verified.

3.2 Informational

3.2.1 Outdated comment for `MAX_BID_PRICE` calculation

Severity: Informational

Context: ContinuousClearingAuction.sol#L45-L47

Description: The documentation comment for `MAX_BID_PRICE` in the `ContinuousClearingAuction` contract contains outdated information about how the maximum bid price is calculated. The comment states that the value is "Set during construction to type(uint256).max / TOTAL_SUPPLY", but the actual implementation uses the `MaxBidPriceLib.maxBidPrice()` library function.

The current comment:

```
/// @notice The maximum price which a bid can be submitted at
/// @dev Set during construction to type(uint256).max / TOTAL_SUPPLY
uint256 public immutable MAX_BID_PRICE;
```

The actual implementation:

```
MAX_BID_PRICE = MaxBidPriceLib.maxBidPrice(TOTAL_SUPPLY);
```

Recommendation: Update the documentation comment to accurately reflect the current implementation:

```
  /// @notice The maximum price which a bid can be submitted at
- /// @dev Set during construction to type(uint256).max / TOTAL_SUPPLY
+ /// @dev Set during construction using MaxBidPriceLib.maxBidPrice() based on
 ↪  TOTAL_SUPPLY
  uint256 public immutable MAX_BID_PRICE;
```

Uniswap Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.2.2 Incorrect step documented in `maxBidPrice` derivation

Severity: Informational

Context: MaxBidPriceLib.sol#L105-L107

Description: In `MaxBidPriceLib.maxBidPrice`, we document the derivation of the `maxBidPrice`, including the following step:

```
*  x = 2 * (154 - y)
* We want to find 2^x, not `x` so we take both sides to the power of 2:
*  2^x = (2^154 / y) ** 2
```

The right side is not correctly computed and instead should be: `(2^155 / 2^y) ** 2` as `y` is not raised to the power of 2 as intended. Note that the actual value used is correct as we are using `_totalSupply`, which is `2^y` in place of `y`.

Recommendation: Adjust the step to raise `y` to the power of 2:

```
   *  x = 2 * (154 - y)
   * We want to find 2^x, not `x` so we take both sides to the power of 2:
-  *  2^x = (2^154 / y) ** 2
+  *  2^x = (2^155 / 2^y) ** 2
```

Uniswap Labs: Acknowledged for now. We'll make an issue to update it later.

Cantina Managed: Acknowledged.