

Coinshift: Chainlink Integration

Security Review

Cantina Managed review by:
Gerard Persoon, Lead Security Researcher
Rvierdiiev, Security Researcher

March 16, 2026

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

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 Prices can only be used from max 1 day old . . . 4
  - 3.1.2 Edge case for `roundId - 1` . . . 4
  - 3.1.3 `answeredInRound` is deprecated . . . 4
- 3.2 Gas Optimization . . . 5
  - 3.2.1 `_disableInitializers()` call in the constructor is redundant . . . 5
  - 3.2.2 Redundant check in `_normalizePrice()` . . . 5
  - 3.2.3 Array length can be cached in `getPriceInfos()` . . . 5
  - 3.2.4 Redundant overflow checks . . . 5
  - 3.2.5 Check in function `_checkPriceBounds()` can be optimized . . . 6
- 3.3 Informational . . . 6
  - 3.3.1 `l2-sequencer-feeds` might be useful . . . 6
  - 3.3.2 Several functions, events and errors are not exposed in `IPricer.sol` . . . 6
  - 3.3.3 Several underlying `init` functions are not called from `initialize()` . . . 7
  - 3.3.4 Unusable deposit-time round forces pricing at a different time . . . 7
  - 3.3.5 Function `setChainlinkFeed()` doesn't emit all changes . . . 7
  - 3.3.6 Impractical values can be set . . . 8
  - 3.3.7 Rapid price changes in combination with frequent oracle updates won't trigger deviations 8
  - 3.3.8 `fallbackDecimals` is never used in `_getCurrentChainlinkResponse()` . . . 8
  - 3.3.9 Function `_getRoundChainlinkResponse()` doesn't check `decimals()` . . . 8
  - 3.3.10 Implicit value `success = false` . . . 9
  - 3.3.11 Similar code `_checkDeviation()` and `_isValidDeviationReference()` . . . 9

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

From Mar 3rd to Mar 5th the Cantina team conducted a review of uspc-contracts on commit hash a07aa38b. The team identified a total of **19** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	3	2	1
Gas Optimizations	5	2	3
Informational	11	4	7
Total	**19**	**8**	**11**

2.1 Scope

The security review had the following components in scope for uspc-contracts on commit hash a07aa38b:

```
src
├── IUSPC.sol
├── IUSPCHub.sol
├── Pricer.sol
├── USPC.sol
└── interfaces
    ├── IAggregatorV3.sol
    ├── IIUSPC.sol
    ├── IIUSPCHub.sol
    ├── IPricer.sol
    └── IUSPC.sol
```

3 Findings

3.1 Low Risk

3.1.1 Prices can only be used from max 1 day old

Severity: Low Risk

Context: IUSPCHub.sol#L412, IUSPCHub.sol#L433, IUSPCHub.sol#L593, IUSPCHub.sol#L603, Pricer.sol#L29, Pricer.sol#L131-L134, Pricer.sol#L139-L147, Pricer.sol#L152-L165, Pricer.sol#L224, Pricer.sol#L234, Pricer.sol#L237, Pricer.sol#L247, Pricer.sol#L365-L367

Description: Prices can only be used from max 1 day old, due to several checks:

The functions `getPrice()`, `getPriceInfo()`, `getPriceInfoscalls()` all call the following functions:

- `_readRound()`.
- `_validateAndNormalize()`.
- `_chainlinkIsFrozen()`.

The function `_chainlinkIsFrozen()` checks if the price has been updated < MAX_PRICE_AGE (e.g. 1 day). This means only prices upto 1 day ago can be retrieved.

Additionally there are checks in `IUSPCHub::processSubscriptions()` and `IUSPCHub::_processSingleRedemption()` which also only allow timestamp upto `MAX_PRICE_AGE` ago.

Recommendation: If older prices are ever relevant (possibly in non standard situations where the contracts have been paused for a day), then consider updating the checks or have a workaround.

Coinshift: Acknowledged. This is intentional behavior, staleness policy is by design. The Hub also independently enforces `MAX_PRICE_AGE` at processing time. Both contracts have the 24 hour window.

Cantina Managed: Acknowledged.

3.1.2 Edge case for `roundId - 1`

Severity: Low Risk

Context: Pricer.sol#L276-L285

Description: In function `_checkDeviation()` the calculation of `currentResponse.roundId - 1` isn't correct if the underlying aggregater has changed. In that situation high bits of `roundId` are updated and the lower bits part of `roundId` will be set to 1.

Recommendation: This situation can be detected by the check `uint64(previousRoundId) == 0`. The most straightforward solution is to revert, because finding the highest `roundId` of the previous aggregater is difficult.

Consider changing the code to:

```
- if (currentResponse.roundId <= 1 || ...) return;
+ if ( uint64(currentResponse.roundId) <= 1 || ... ) return; // strip the high bits
```

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.1.3 `answeredInRound` is deprecated

Severity: Low Risk

Context: Pricer.sol#L237-L246, Pricer.sol#L323, Pricer.sol#L344, Pricer.sol#L351-L363

Description: The value of `answeredInRound` is deprecated, see the chainlink documentation. Some oracles might return a value of 0, which will result in a revert during the check.

Recommendation: Consider ignoring `answeredInRound`.

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.2 Gas Optimization

3.2.1 `_disableInitializers()` call in the constructor is redundant

Severity: Gas Optimization

Context: Pricer.sol#L67-L69

Description: `_disableInitializers()` call in the constructor can be removed as the same code exists inside the constructor of `AccessControlledUUPSUpgradeable` parent contract.

Recommendation: Remove redundant functionality.

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.2.2 Redundant check in `_normalizePrice()`

Severity: Gas Optimization

Context: Pricer.sol#L26, Pricer.sol#L259-L262

Description: The check for `decimals == TARGET_DECIMALS` is very unlikely to trigger in practice because `decimals` usually are 8 or 18, which is smaller than `TARGET_DECIMALS`. Without this statement, the code will still function correctly and some gas will be saved.

Recommendation: Consider removing the statement:

```
- if (decimals == TARGET_DECIMALS) return rawPrice;
```

Coinshift: Acknowledged. We will keep the check, as removing the defensive guard doesn't save significant gas.

Cantina Managed: Acknowledged.

3.2.3 Array length can be cached in `getPriceInfos()`

Severity: Gas Optimization

Context: Pricer.sol#L152-L165

Description: In the `for` loop, the value of `_priceIds.length` is calculated for every cycle. This can be optimized.

Recommendation: Consider caching the value of `_priceIds.length`.

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.2.4 Redundant overflow checks

Severity: Gas Optimization

Context: Pricer.sol#L263, Pricer.sol#L298

Description: There are a few checks that prevent overflows during multiplications. Solidity will also catch these types of errors so the checks aren't really necessary. Removing them will save some gas.

Recommendation: Consider removing the checks.

Coinshift: Acknowledged. Custom errors are intentionally used because they are clearer than a raw panic for debugging and monitoring.

Cantina Managed: Acknowledged.

3.2.5 Check in function `_checkPriceBounds()` can be optimized

Severity: Gas Optimization

Context: Pricer.sol#L200-L201, Pricer.sol#L210-L213, Pricer.sol#L267-L274

Description: Function `_checkPriceBounds()` every time checks `minPrice == 0 || maxPrice < minPrice`. This checks can be done one time during configuration which will save same gas.

Recommendation: Consider moving this check to `setBoundsCheckEnabled()`.

> **Note:** the check is already present in `setMaxPriceDeviationBps()`.

```
  function _checkPriceBounds(...) ... {
  //...
-     if (minPrice == 0 || maxPrice < minPrice) revert InvalidPriceBounds();
  //...
  }
  function setBoundsCheckEnabled(...) ... {
+     if (minPrice == 0 || maxPrice < minPrice) revert InvalidPriceBounds(); // have to
↪  be set correctly first
  //...
  }
```

Coinshift: Acknowledged. It protects against the case where bounds are set and then are somehow corrupted then enabled. Gas costs are negligible in this case.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 `l2-sequencer-feeds` might be useful

Severity: Informational

Context: Pricer.sol#L15

Description: The uptime of the L2 isn't taken into account in the `Pricer` contract. The project indicated that the current plans are to deploy on ethereum mainnet.

Recommendation: If the protocol will be deployed on a L2, doublecheck the usefulness of the `l2-sequencer-feeds`.

Coinshift: Acknowledged. We will be deploying on eth mainnet. L2 sequencer uptime checks are not applicable.

Cantina Managed: Acknowledged.

3.3.2 Several functions, events and errors are not exposed in `IPricer.sol`

Severity: Informational

Context: Pricer.sol#L58-L62, Pricer.sol#L64, Pricer.sol#L95, Pricer.sol#L99, Pricer.sol#L177, Pricer.sol#L369

Description: Several functions, events and errors are not exposed in `IPricer.sol`. This makes interfacing with them more difficult.

Recommendation: Consider add the function definitions to `IPricer.sol` and moving the errors to `IPricer.sol`.

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.3.3 Several underlying `init` functions are not called from `initialize()`

Severity: Informational

Context: Pricer.sol#L75-L78

Description: Several underlying `init` functions are not called from `initialize()`. The functions are currenlty empty, but that might be changed in the future.

Recommendation: Consider also calling:

- `__ERC165_init()`.
- `__Context_init()`.

Coinshift: Achknowledged. These are currently empty no ops. We'll review init chains when upgrading oz dependencies.

Cantina Managed: Acknowledged.

3.3.4 Unusable deposit-time round forces pricing at a different time

Severity: Informational

Context: IUSPCHub.sol#L422

Description: Deposits and redemptions are processed with a Chainlink round id(`priceId`) chosen to match the time of the operation.

If that round fails validation (e.g. `DeviationTooHigh`, `StalePriceData`, `PriceNotFound`), it is not possible to use it for that operation:

- The operation cannot be processed with the deposit-time priceId.
- The only way to process it is for the admin to call `processSubscriptions` / `processRedemptions` with a different priceId - i.e. a different Chainlink round, which corresponds to a different time than when the deposit (or redemption) actually occurred.

So when the correct round for the deposit/redemption time is unusable, the admin is forced to price the user at another moment in time, which can be seen as a temporal mismatch or fairness concern (user is minted/redeemed at a price from a time other than their operation time).

Recommendation: Consider documenting that if the round at deposit/redemption time cannot be used (e.g. due to deviation or staleness), the operation can only be processed by using another round.

Coinshift: Acknowledged. This is Intentional behavior. The system should halt rather than process at stale prices. Adding an admin override increases attack surface for an extreme scenario.

The admin gathers `roundIds` off-chain, if any `roundId` fails validation the batch reverts and admin resubmits with valid `roundIds`.

Cantina Managed: Acknowledged.

3.3.5 Function `setChainlinkFeed()` doesn't emit all changes

Severity: Informational

Context: Pricer.sol#L177-L190

Description: The function `setChainlinkFeed()` doesn't emit all changes.

Recommendation: Consider also emitting the old and new `decimals`.

Coinshift: Acknowledged. Feed decimals are queryable on chain from the feed contract directly. Not worth changing the event signature for data that is already accessible.

Cantina Managed: Acknowledged.

3.3.6 Impractical values can be set

Severity: Informational

Context: Pricer.sol#L192-L208

Description: Impractical values can be set, which will limit any changes to the prices.

- `maxPriceDeviationBps` can be set to 0.
- `minAllowedPrice` can be set to the same values as `maxAllowedPrice`.

Recommendation: Consider disallowing to set impractical values.

Conshift: Acknowledged. It is on a trust based system, admin is trusted with role gated access.

Cantina Managed: Acknowledged.

3.3.7 Rapid price changes in combination with frequent oracle updates won't trigger deviations

Severity: Informational

Context: Pricer.sol#L299-L300

Description: When rapid price changes occur and the oracle is frequently updated the price difference between two updates can be small enough to not trigger deviations. Also when the price flattens out the prices difference will certainly be low between oracle updates.

Recommendation: Decide what actions to take during and direct after rapid price changes. Consider taking the time between oracle updates into account.

Coinshift: Acknowledged. This behaviour is expected, time weighting will add significant complexity.

Cantina Managed: Acknowledged.

3.3.8 `fallbackDecimals` is never used in `_getCurrentChainlinkResponse()`

Severity: Informational

Context: Pricer.sol#L303-L314

Description: The `try/catch` in function `_getCurrentChainlinkResponse()` directly returns. This means the transaction will directly revert and `fallbackDecimals` will never be used.

Recommendation: Doublecheck the intended action. If this is intentional then consider removing `fallbackDecimals`.

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.3.9 Function `_getRoundChainlinkResponse()` doesn't check `decimals()`

Severity: Informational

Context: Pricer.sol#L177-L182, Pricer.sol#L303-L314, Pricer.sol#L330-L337

Description: Function `_getRoundChainlinkResponse()` doesn't check `decimals()`, while function `_getCurrentChainlinkResponse()` does perform this check. This is inconsistent.

Recommendation: If you want anticipate a change in `decimals()` then it should be implemented consistently.

And to be fully consistent the following check, which is also done in `setChainlinkFeed()`, could also be added:

```
+ if (decimals > TARGET_DECIMALS) revert UnsupportedFeedDecimals();
```

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.3.10 Implicit value `success = false`

Severity: Informational

Context: Pricer.sol#L303, Pricer.sol#L312-L314, Pricer.sol#L324-L325, Pricer.sol#L325, Pricer.sol#L330, Pricer.sol#L345-L346

Description: The functions `_getCurrentChainlinkResponse()` and `_getRoundChainlinkResponse()` leave the value of `response.success` in their default value, which is `false`. This values is interpreted by the caller of the function to know the function has failed. However this behaviour might not be obvious for readers of the code.

Recommendation: Consider adding a comment:

```
- catch { ... }
+ catch { ... } // leave response.success == false
```

Coinshift: Fixed in PR 14.

Cantina Managed: Fix verified.

3.3.11 Similar code `_checkDeviation()` and `_isValidDeviationReference()`

Severity: Informational

Context: Pricer.sol#L237-L247, Pricer.sol#L276-L289, Pricer.sol#L351-L363

Description: The checks in `_checkDeviation()` and `_isValidDeviationReference()` are very similar. Because the checks are not trivial, having it in two places makes maintenance of the code more difficult.

Recommendation: Consider combining the checks. The check for `_chainlinkIsFrozen()` could also be added.

Coinshift: Acknowledged. It is doing different error handling:

- `_validateAndNormalize()` reverts on bad data.
- `_isValidDeviationReference()` returns false (skip).

Combining them into one method with a revert-or-skip flag reduces clarity without improving safety.

Cantina Managed: Acknowledged.