Size Credit
Security Review

DECEMBER, 2024

www.chaindefenders.xyz
https://x.com/ChDefendersEth

Lead Auditors

PeterSR

0x539.eth

Table of Contents

Protocol Summary

Size is a credit marketplace with unified liquidity across maturities.

In this version v1.5.1, Size addresses a known limitation of the protocol - that of not having a fallback oracle in case Chainlink goes down.

They have decided to use Uniswap v3 TWAP as a fallback oracle.

Disclaimer

The ChainDefenders team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

Likelihood/Impact	High	Medium	Low
High	H	H/M	M
Medium	H/M	M	M/L
Low	M	M/L	L

Audit Details

Scope

Id	Files in scope
1	IPriceFeed.sol
2	IPriceFeedV1_5_1.sol
3	PriceFeed.sol
4	ChainlinkPriceFeed.sol
5	ChainlinkSequencerUptimeFeed.sol
6	UniswapV3PriceFeed.sol

Roles

Id	Roles
1	DEFAULT_ADMIN_ROLE
2	KEEPER_ROLE
3	BORROW_RATE_UPDATER_ROLE
4	PAUSER_ROLE

Executive Summary

Issues found

Severity	Count	Description
High	0	Critical vulnerabilities
Medium	0	Significant risks
Low	1	Minor issues with low impact
Informational	2	Best practices or suggestions
Gas	2	Optimization opportunities

Findings

Low

Low 01 Faulty Cardinality Assumption

Location

UniswapV3PriceFeed.sol:33

Description

In the `UniswapV3PriceFeed` the desired cardinality is calculated by dividing the TWAP window by the average block time. This average block time is provided as a variable in the constructor on deployment.

While such calculation has logic (to contain all observations of a given TWAP window) it is important to note that it has some pitfalls:

1. Average block time is a changing variable which can vary depending on the network conditions;
2. If the TWAP window is not big enough the calculation can return a small number making the price feed even more vulnerable to manipulation.

Recommendation

To solve this issue we recommend:

1. Add a minimum desired cardinality and enforce it:

```
constructor(
    uint256 _decimals,
    IERC20Metadata _baseToken,
    IERC20Metadata _quoteToken,
    IUniswapV3Factory _uniswapV3Factory,
    IUniswapV3Pool _pool,
    uint32 _twapWindow,
-   uint32 _averageBlockTime
+   uint32 _averageBlockTime,
+   uint16 minimumDesiredCardinality
) {
    if (
        address(_baseToken) == address(0) || address(_quoteToken) == address(0)
            || address(_uniswapV3Factory) == address(0) || address(_pool) == address(0)
    ) {
        revert Errors.NULL_ADDRESS();
    }
    if (address(_baseToken) == address(_quoteToken)) {
        revert Errors.INVALID_TOKEN(address(_quoteToken));
    }
    if (_twapWindow == 0) {
        revert Errors.INVALID_TWAP_WINDOW();
    }
    if (_averageBlockTime == 0) {
```

```
25          revert Errors.INVALID_AVERAGE_BLOCK_TIME();
26      }
27
28      decimals = _decimals;
29      uniswapV3Factory = _uniswapV3Factory;
30      baseToken = _baseToken;
31      quoteToken = _quoteToken;
32      pool = _pool;
33      twapWindow = _twapWindow;
34      averageBlockTime = _averageBlockTime;
35
36      // slither-disable-next-line unused-return
37      (,,, uint16 cardinality,,,) = IUniswapV3Pool(_pool).slot0();
38      uint16 desiredCardinality = SafeCast.toUint16(FixedPointMathLib.
        divUp(_twapWindow, _averageBlockTime) + 1);
39  +   if (desiredCardinality < minimumDesiredCardinality) {
40  +       desiredCardinality = minimumDesiredCardinality;
41  +   }
42      if (cardinality < desiredCardinality) {
43          pool.increaseObservationCardinalityNext(desiredCardinality);
44      }
45  }
```

2. Carefully choose TWAP window based on the markets you currently have:

The choice of a time-weighted average price (TWAP) window in Uniswap v3 (or any decentralized exchange) depends on the trade-off between market responsiveness and manipulation resistance. Here's how TWAP windows are typically decided:

Shorter TWAP Windows (e.g., a few minutes):

- Pros: Capture rapid price changes, making them more reflective of the current market.
- Cons: More susceptible to price manipulation, especially in low-liquidity pools or during periods of high volatility.

Longer TWAP Windows (e.g., hours or days):

- Pros: Provide better resistance to manipulation since sustained price distortion is expensive. These are suitable for protocols needing price stability over time.
- Cons: Respond slower to market changes, which may not be ideal for dynamic strategies or highly volatile assets.

Popular Choices:

Many implementations use windows like 10 minutes to several hours for TWAP calculations to balance security and responsiveness. A commonly cited example is a 24-hour TWAP for applications needing strong manipulation resistance.

Factors to Consider:

- Liquidity of the trading pair: Higher liquidity reduces manipulation risk, allowing for shorter windows.
- Purpose of the TWAP: Price-sensitive applications (e.g., collateral calculations) might prefer longer windows for stability, while trading-focused apps might favour shorter windows for responsiveness.
- Historical price volatility: More volatile pairs may need longer TWAPs to smooth out sharp price movements.

Status

Acknowledged.

Response: This piece of code was introduced to avoid having to manually update the cardinality of a Uniswap v3 pool during the deployment of the UniswapV3PriceFeed oracle, as some implementations do (such as Euler's Price Oracle: here), which can be useful in cases of factory deployments. Since this resize of the cardinality of the observation buffer is performed only once, the deployer would set `_averageBlockTime` according to the value at the time of initialization. If this value is lower than desired, the deployer can manually update the buffer if needed, which would be similar to not having increaseObservationCardinalityNext called on the contract at all. In any case, the recommendations for TWAP size are appreciated.

Informational

Info 01 `uniswapV3Factory` Is Never Used

Location

UniswapV3PriceFeed.sol:28

Description

`uniswapV3Factory` is passed to the constructor of `PriceFeed` as parameter part of `PriceFeedParams memory priceFeedParams` and then populated to `UniswapV3PriceFeed`, but factory is not used in `UniswapV3PriceFeed`.

```
constructor(PriceFeedParams memory priceFeedParams) {
  chainlinkSequencerUptimeFeed = new ChainlinkSequencerUptimeFeed(
   priceFeedParams.sequencerUptimeFeed);
  chainlinkPriceFeed = new ChainlinkPriceFeed(
      decimals,
      priceFeedParams.baseAggregator,
      priceFeedParams.quoteAggregator,
      priceFeedParams.baseStalePriceInterval,
      priceFeedParams.quoteStalePriceInterval
  );
  uniswapV3PriceFeed = new UniswapV3PriceFeed(
      decimals,
      priceFeedParams.baseToken,
      priceFeedParams.quoteToken,
      priceFeedParams.uniswapV3Factory,
      priceFeedParams.pool,
      priceFeedParams.twapWindow,
      priceFeedParams.averageBlockTime
  );
}
```

Recommendation

Consider removing this variable, because it's never used.

Status

Fixed

Info 02 Missing Documentation In `IPriceFeedV1_5`

Location

IPriceFeedV1_5.sol

Description

Currently all methods in the interface `IPriceFeedV1_5` are left without documentation.

Recommendation

Include documentation to the `IPriceFeedV1_5` for all of the methods.

Status

Fixed

Gas

Gas 01 StartedAt Should Not Be Checked Because It Never Returns 0 Except On Arbitrum

Location

ChainlinkSequencerUptimeFeed.sol:31

Description

When calling `validateSequencerIsUp` to validate that the sequencer is up a call to `latestRoundData` is made. Two of the returned values are `startedAt` and `answer`.

An issue in the Code4rena's contest was validated stating that `startedAt` can return 0 when a round is invalid and that's why it is important to include such a check to prevent wrong calculations. However, the Chainlink documentation states that `startedAt` can never be equal to 0 except on Arbitrum:

> The startedAt variable returns 0 only on Arbitrum when the Sequencer Uptime contract is not yet initialized. For L2 chains other than Arbitrum, startedAt is set to block.timestamp on construction and startedAt is never 0.After

> the feed begins rounds, the startedAt timestamp will always indicate when the sequencer feed last changed status.

For more information you can find the documentation -> here

Recommendation

Having this in mind (and the fact that the protocol is deployed on Base) our recommendation is to remove this check as it spends ~20 more gas on every call.

Status

Acknowledged.

Response: We prefer to keep the redundancy in case the protocol expands and deploys to other networks, such as Arbitrum.

Gas 02 Instead Of Marking A Function `public`, Consider Marking It `external` If It Is Not Used Internally

Location

UniswapV3PriceFeed.sol:74

Description

The function `getPrice` in UniswapV3PriceFeed.sol is currently marked as public, even though it is not used internally within the contract. Using the public visibility unnecessarily allows internal access while being less gas-efficient compared to external for externally called functions.

Recommendation

Change the identifier from `public` to `external`

Status

Fixed