Fixed Point Solutions, LLC

Space AMM Oracle Implementation Assessment

2022/03/15
Prepared by: Kurt Barry

1. Scope

The following PR, which implemented a Balancer TWAP oracle for the Space AMM, among other changes, was reviewed:
https://github.com/sense-finance/space-v1/pull/8

Fixes for the issues raised as well as some later changes were reviewed. Due to PRs being split and rebased, it is unfortunately difficult to determine the exact set of commits in the space-v1 repo that were reviewed; findings link to a specific commit that contains the finding, and fixes refer to a specific commit that resolved the finding. It may be assumed that any referenced commits were reviewed.

2. Limitations

No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3. Findings

Findings and recommendations are listed in this section, grouped into broad categories. It is up to the team behind the code to ultimately decide whether the items listed here qualify as issues that need to be fixed, and whether any suggested changes are worth adopting. When a response from the team regarding a finding is available, it is provided.

Findings are given a severity rating based on their likelihood of causing harm in practice and the potential magnitude of their negative impact. Severity is only a rough guideline as to the risk an issue presents, and all issues should be carefully evaluated.

Severity Level Determination		Impact		
		High	Medium	Low
Likelihood	High	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**

Issues that do not present any quantifiable risk (as is common for issues in the Code Quality category) are given a severity of **Informational**.

3.1 Security and Correctness

Findings that could lead to harmful outcomes or violate the intentions of the system.

SC.1 Error In Computation of Protocol Fee

Severity: Low

Code Location:
https://github.com/sense-finance/space-v1/blob/6c687021a16921207a262b44e9c28eebe3133f2e/src/Space.sol#L495

Description: To determine the growth in balances, the ratio of the YieldSpace invariant at the current time to what its value would have been if there had been no trades since its last measurement is used. However, this quantity is not directly proportional to the token balances; rather it is proportional to the balances raised the the power 1 - t (where t is the time-to-maturity of the pool). This will systematically underestimate the actual growth and thus the collected protocol fees will be smaller than they should be. This can be corrected by raising the ratio to the power of 1/(1 - t).

Response: Fixed in commit 1d3df05cf7de02b38a3abdbc386f912da8c340c2.

SC.2 Biased Price Sampling

Severity: Low

Code Location:
https://github.com/sense-finance/space-v1/blob/6c687021a16921207a262b44e9c28eebe3133f2e/src/Space.sol#L553

Description: The TWAP implementation samples the current swap price by simulating a small swap of Principal Tokens (PTs) to Target. This calculation is subject to both a very small amount of slippage (limited by swapping a quantity small compared to the liquidity present), and more importantly, the swap fee (unaffected by swap size). Both effects introduce a consistent bias towards a lower PT price in Target terms.

The accuracy of the price measurement could be increased in at least two ways. The first option would be to temporarily reduce the fee to zero before executing `_onSwap()` to sample the price, then resetting it to its previous level afterwards. This is straightforward but increases gas costs. The second option would be to obtain the instantaneous price by leveraging formula (2) from the YieldSpace paper (https://yield.is/YieldSpace.pdf) to compute the implied interest rate, and then using that interest rate and the current time-to-maturity to calculate the swap price. This eliminates bias from both slippage and fees, and would likely be even more gas-efficient than the current implementation.

Response: Fixed in commit 1d3df05cf7de02b38a3abdbc386f912da8c340c2. See finding SC.3 for a note on an issue with the fix.

SC.3 `getImpliedRateFromPrice` and `getPriceFromImpliedRate` Use the Wrong Time Constant

Severity: Low

Code Location:
[1]https://github.com/sense-finance/space-v1/blob/1d3df05cf7de02b38a3abdbc386f912da8c340c2/src/Space.sol#L626
[2]https://github.com/sense-finance/space-v1/blob/1d3df05cf7de02b38a3abdbc386f912da8c340c2/src/Space.sol#L639

Description: These functions should use the time constant of the Space pool (`ts`), rather than a time period of one year, in their calculations. In particular, the use of `getImpliedRateFromPrice` in `getFairBPTPrice` will lead to a wrong result as it calculates the time-to-maturity value for deriving the pool invariant using `ts`. The prices recorded in `_updateOracle`, which uses `getPriceFromImpliedRate`, will be incorrect as well. (The exception to both, of course, is the case that `ts` is exactly one year.)

Response: Fixed in commit 764a9e6f34a9661551e310bfcdb8fb4321a65f37.

SC.4 Incorrect Maximum Safe Query Window Value

Severity: Low

Code Location:
https://github.com/sense-finance/space-v1/blob/cb58702ec5da093420b82338693961b54f4fcd81/src/oracle/PoolPriceOracle.sol#L105

Description: The value is hardcoded as 5.5 hours, but the correct value is `_MAX_SAMPLE_DURATION` (20 minutes) times `Buffer.SIZE` (20), or 400 minutes (6.66… hours).

Response: Fixed in commit 8d7779474d6e3ff9f74487340c7bf91b6cba4318.

3.2 Usability and Incentives

Findings that could lead to suboptimal user experience, hinder integrations, or lead to undesirable behavioral outcomes.

U.1 Price Sampling Does Not Follow Balancer Conventions

Severity: Informational

Code Location:
https://github.com/sense-finance/space-v1/blob/6c687021a16921207a262b44e9c28eebe3133f2e/src/Space.sol#L561

Description: The price samples passed to Balancer's oracle logic always express the price of the Principal Token in terms of the Target; Balancer's own documentation states that the price should be that of the second token expressed in units of the first as an 18-decimal fixed point value:

https://github.com/balancer-labs/balancer-v2-monorepo/blob/c40b9a783e328d817892693bd13b4a14e4dcff4d/pkg/pool-utils/contracts/interfaces/IPriceOracle.sol#L31 .

It is possible that this departure will reduce composability as Space's price oracle will not work "out of the box" with other protocols or tools built around Balancer's oracles. On the other hand, the chosen convention may be easier to reason about for integrators and provide a higher degree of consistency among different Space pools, so no definite recommendation is made here.

Response: Switched to the Balancer convention in commit 1d3df05cf7de02b38a3abdbc386f912da8c340c2.

3.3 Gas Optimizations

Findings that could reduce the gas costs of interacting with the protocol, potentially on an amortized or averaged basis.

G.1 Code Locations Still Using `getIndices()`

Severity: Low

Code Location:
[1]https://github.com/sense-finance/space-v1/blob/3821cbeb6127fd3a0d8c0972d99b01873de0e006/src/Space.sol#L186
[2]https://github.com/sense-finance/space-v1/blob/3821cbeb6127fd3a0d8c0972d99b01873de0e006/src/Space.sol#L386

Description: These can be replaced with the more gas-efficient `pti` / `1 - pti` pattern.

Response: Fixed in commit 6c687021a16921207a262b44e9c28eebe3133f2e.

3.4 Code Quality

None.

4. Notes

This section contains general considerations for interacting with or maintaining the system and various conclusions reached or discoveries made during the course of the assessment. Whereas findings generally represent things for the team to consider changing, notes are more informational and may be helpful to those who intend to interact with the system.

4.1 Balancer Oracles Only Sample One Price Per Block

Balancer TWAP oracles, as an implementation constraint, are only able to sample a single price per block. If many trades happen within a block, and if they mostly move the price in one direction, the stored price may in fact be a poor representation of the price during that block (especially if the first trade is small and doesn't move the price very much).

More theoretically, one can imagine a multi-block MEV strategy where if an entity can consistently land the final transaction in one block and the first transaction in the next, that entity could manipulate the price in the oracle. E.g.

```
Block N__________
....
borrow funds + swap to desired price
```

```
________________
/\
||
Block N+1________
small swap at desired price + swap fully back to old price + repay borrow
...
________________
```

Such a strategy is not risk free and comes with swap and borrow costs, but as MEV extraction becomes more sophisticated, especially with the transition to proof-of-stake altering the current competitive equilibrium, more such exotic attacks may be attempted.

The mitigation for such concerns in practice is likely to ensure that sufficiently long periods are being sampled such that the cost of manipulation is prohibitive (although this comes with some trade-off in price accuracy).