Arrakis V2 Manager Templates

Prepared by: Vaibhav Sutar

Table of Contents

Disclaimer

The **Vaibhav Sutar** audit 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

Severity	Description
High	Bugs leading to assets theft, fund access locking, or any other loss of funds to be transferred to any party.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds.
Low	Low impact and low/medium likelihood events where assets are not at risk.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Audit Details

The findings described in this document correspond the following commit hash:

```
1d507a2beaa9c0e785bac7dd943c77964fedaef3
```

Scope

```
├── contracts
│   ├── SimpleManager.sol
│   └── Oracles
│       ├── ChainLinkOracle.sol
|       └── UniswapV3PoolOracle.sol
```

Executive Summary

Issues found

Severity	Findings
Critical	0
High	0
Medium	0
Informational	6
Total	**6**

Tools Used

- Slither
- Aderyn
- cloc
- Solidity Metrics (audit estimation)
- Solidity Visual Developer

Findings

Medium

[M-01]: Price Calculation Overflow in Rebalance

Description:

Squaring sqrtPriceX96 in the price calculation can overflow for very high values (>= 2^128), causing the rebalance function to revert and become unusable for imbalanced pools. This creates a denial-of-service condition.

This issue also affects the following functions:

- Twap.sol#L45
- Twap.sol#L60

Recommendation:

Implement overflow protection similar to the `getQuoteAtTick` function from Uniswap V3's Quote library.

[M-02] :Chainlink oracle not handled correctly

Description

- The ChainLinkOracle contract has problems in the functions getPrice0 and getPrice1 because latestRoundData() is not handled properly.
- There is no check to make sure updatedAt is recent. Without this, the contract might use old or unreliable price data.
- The code does not support Layer 2 networks that need a Sequencer contract for correct data.
- There is no try-catch block. If the price feed reverts, the whole transaction fails.

Recommendation

- To fix these issues:
- Add a check to make sure updatedAt is recent and ignore old data.
- Update the oracle logic to support Layer 2 networks that use a Sequencer contract.

- Add a try-catch block so that a revert from the price feed does not stop the whole transaction. You can also add a backup oracle (like Uniswap V3) to use if the main oracle fails.

[M-03]:Rounding error in ChainLinkOracle

Description

- In the getPrice0 function there is a rounding mistake. The code divides too early, so small values get rounded down before they are multiplied. This makes the final price wrong. For example, when priceFeedDecimals < token1Decimals, the result can be smaller than it should be in some cases it even becomes 0. The same mistake is in getPrice1.

```
FullMath.mulDiv(
   FullMath.mulDiv(
       10 ** priceFeedDecimals,
       10 ** priceFeedDecimals,
       SafeCast.toUint256(price)
   ),
   10 ** token1Decimals,
   10 ** priceFeedDecimals
);
```

The code does a division before a multiplication, so precision is lost. You should multiply first, then divide.

Recommendation

- Change the math so you multiply before you divide. Use FullMath.mulDiv to keep precision. For example, do the inner multiplication with larger powers first, then divide:

```
// multiply first, then divide to avoid rounding errors
FullMath.mulDiv(
  FullMath.mulDiv(
    10 ** (2 * priceFeedDecimals),
    10 ** token1Decimals,
    SafeCast.toUint256(price)
  ),
  1,
  10 ** priceFeedDecimals
);
```

- This fixes the rounding error in both getPrice0 and [getPrice1].

Informational

[I-01]: Unused Imports in `SimpleManager`

Description:

- Line :

1. SimpleManager.sol#L8
2. SimpleManager.sol#L23

- The `ERC20` import is unused.
- The `Range` import is unnecessary, as it's already accessible through the `Rebalance` struct

Recommendation

- Remove the unused `ERC20` and redundant `Range` imports to improve code clarity and maintainability.

[I-02]: Unused function in `SimpleManager`.

Description:

- Line : SimpleManager.sol#L357
- Function `_includeAddress` is unused in SimpleManager

Recommendation

- Remove the unused function.

[I-03]: Some function lack zero address checks.

Description:

Some function lack zero address check for their input parameters. The affected functions are:

- SimpleManager.sol#L73
- SimpleManager.sol#L77
- ChainLinkOracle.sol#L23
- UniswapV3PoolOracle.sol#L16

Recommendation

Add zero checks in listed functions .

[I-04] : Gas Optimization

Description

1. Use ++i instead of i++

- SimpleManager.sol#L143
- SimpleManager.sol#L178
- SimpleManager.sol#L213
- SimpleManager.sol#L223
- SimpleManager.sol#L236
- SimpleManager.sol#L251
- SimpleManager.sol#L348

1. Delete "dead" code

- SimpleManager.sol#L357-L370

3. Remove a useless check

- SimpleManager.sol#L208
- SimpleManager.sol#L269
- SimpleManager.sol#L347

Recommendation

It is recommended to optimize the following areas of code .

[I-05]: No checks if vault is in vaults

Description

No checks if vault is in vaults

1. In the function SimpleManager.rebalance(), an operator can burn liquidity from a vault in which SimpleManager is the manager, but it initManagement() was not called yet.
2. In the function SimpleManager.withdrawAndCollectFees(), the owner can withdraw manager balance from a vault in which SimpleManager is the manager, but it initManagement() was not called yet.

Recommendation

- We recommend adding checks if vault is in vaults

[I-06]: Repeated Code

Description

- In the rebalance() function, the oracle is called twice to get the price of token0: once inside the if condition and once inside the _checkMinReturn function. This double call is not needed. You can avoid it by getting the oracle price once, outside the if condition, and then passing that value to the internal function. The price should always be checked anyway.

Recommendation

- We suggest removing the double call. This will be easier if simpleManager always checks the difference between the pool price and the oracle price.

[I-07]: Max deviation has no upper limit.

Description

- In the SimpleManager.initManagement() function, the value params.maxDeviation has no upper limit. If the vault owner sets this value incorrectly, an operator could misuse it and sandwich the rebalance by swapping before and after it.

Recommendation

- We suggest adding a maximum allowed value for params.maxDeviation.

[I-08]: No way to remove or update vault info

Description

- The SimpleManager.initManagement() function adds a vault and sets its info. But after this, there is no way to remove the vault or change its info. We suggest adding a function that lets the SimpleManager owner remove a vault. If vault info needs to be changed, the owner could remove the vault, and then the vault owner could add it again with new settings. You should also think about the risk of vault owners trying to frontrun the rebalance() function.

Recommendation

- We recommend adding a function that allows removing a vault from the list.