STATE
MIND

Arrakis v2 core (rev. 2)

06-04-2023 - 29-08-2023

Table of contents

1. Project Brief

Title	Description
Client	Arrakis
Project name	Arrakis v2 core (rev. 2)
Timeline	06-04-2023 - 29-08-2023
Initial commit	a9759d1a45bc3a9dc9a378cbff3588e76a5083f5
Final commit	f0200abcd73ce994b0641b7cd0e8bc4e2fbcb818

Short Overview

Arrakis is web3's liquidity layer, which at its core acts as a decentralized market-making platform enabling projects to create deep liquidity for their tokens on decentralized exchanges.

The core contracts allow users to:

- Create an ArrakisV2 vault instance that manages holdings of a given token pair
- Dispatch and collect these holdings to/from Uniswap V3 Liquidity Positions (for the defined token pair) via a settable manager smart contract
- Configure important vault setup parameters (manager, restrictedMint, pools) via the vault owner role

Project Scope

The audit covered the following files:

- ArrakisV2Storage.sol
- ArrakisV2Helper.sol
- Underlying.sol
- ArrakisV2Resolver.sol
- ArrakisV2.sol
- ArrakisV2Factory.sol

2. Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

3. Summary of findings

Severity	# of Findings
Critical	0 (0 fixed, 0 acknowledged)
High	0 (0 fixed, 0 acknowledged)
Medium	2 (2 fixed, 0 acknowledged)
Informational	7 (6 fixed, 1 acknowledged)
Total	9 (8 fixed, 1 acknowledged)

4. Conclusion

During the audit of Arrakis v2 core codebase, 9 issues were found in total:

- 2 medium severity issues (2 fixed)
- 7 informational severity issues (6 fixed, 1 acknowledged)

The final reviewed commit is f0200abcd73ce994b0641b7cd0e8bc4e2fbcb818.

Contracts are deployed on ethereum, arbitrum, base, bsc, optimism, polygon networks under the same addresses.

Deployment

File name	Contract deployed on mainnet
ArrakisV2.sol	0x7f346f1eb7a65ff83f51b3fd76dcc70979e6df38
ArrakisV2Factory.sol	0xf90aaFaBb1A4C0CE318Be12Da73F0f31FaBE865d
ArrakisV2Helper.sol	0x89E4bE1F999E3a58D16096FBe405Fc2a1d7F07D6
ArrakisV2Resolver.sol	0x535C5fDf31477f799366DF6E4899a12A801cC7b8

Underlying.sol	0x39b9891bA3C5a8Fe69c19F54Db2fd90A483B780A

5. Findings report

MEDIUM-01	No checks if liquidity to mint is zero	Fixed at 37c595

Description

1. In the function **ArrakisV2.mint()**, if the **mintAmount_** is small then liquidity can round down to **0**. In that case, the transaction reverts in **pool.mint()**. It is recommended to move the check at the line ArrakisV2.sol#L136 to line 141.

```
uint128 liquidity = Position.getLiquidityByRange(
    pool,
    me,
    range.lowerTick,
    range.upperTick
);
if (liquidity == 0) continue;

liquidity = SafeCast.toUint128(
    FullMath.mulDiv(liquidity, mintAmount_, ts)
);

pool.mint(me, range.lowerTick, range.upperTick, liquidity, "");
```

2. In the function **ArrakisV2.rebalance()**, there is no check if **rebalanceParams_.mints[i].liquidity == 0**.

```
(uint256 amt0, uint256 amt1) = IUniswapV3Pool(pool).mint(
    address(this),
    rebalanceParams_.mints[i].range.lowerTick,
    rebalanceParams_.mints[i].range.upperTick,
    rebalanceParams_.mints[i].liquidity,
    ""
);
```

3. In the function **ArrakisV2Resolver.standardRebalance()**, there is no check if **rebalanceParams.mints[i].liquidity == 0**.

```
uint128 liquidity = LiquidityAmounts.getLiquidityForAmounts(
    sqrtPriceX96,
    TickMath.getSqrtRatioAtTick(rangeWeight.range.lowerTick),
    TickMath.getSqrtRatioAtTick(rangeWeight.range.upperTick),
    FullMath.mulDiv(amount0, rangeWeight.weight, hundredPercent),
    FullMath.mulDiv(amount1, rangeWeight.weight, hundredPercent)
);

rebalanceParams.mints[i] = PositionLiquidity({
    liquidity: liquidity,
    range: rangeWeight.range
});
```

Recommendation

We recommend to move the check for the first case and add checks for cases 2 and 3.

Client's comments

For M-1 issue I only changed in the core and the resolver still could add a liquidity==0 to the mint array of rebalance struct, however this will no longer cause a revert during rebalance since rebalance was altered in the core to simply continue for any mints where liquidity==0.

MEDIUM-02	First minter can skew the initial ratio	Fixed at 6cdb18

Description

In the function **ArrakisV2.mint()**, it is possible to skew the initial ratio in a certain case:

```
// inits
init0M = 1e6
init1M = 1e18
mintAmount = 1

// then because the division is rounded up
amount0 = 1
amount1 = 1
amount0Mint = 1e12
amount1Mint = 1

// the check passes since
min(amount0Mint, amount1Mint) == mintAmount
min(1e12, 1) == 1
```

The minimum **mintAmount** in this case should be **1e12**.

Recommendation

We recommend to additionally check if **mintAmount * init / denominator == 0**:

```
if (FullMath.mulDiv(mintAmount_, init0M, denominator) == 0) {
    amount0 = 0;
}
if (FullMath.mulDiv(mintAmount_, init1M, denominator) == 0) {
    amount1 = 0;
}
```

INFORMATIONAL-01	getAmountsForDelta reverts for liquidity < 0	Fixed at 37c595

Description

The **Underlying** library has a function **getAmountsForDelta** that accepts a **liquidity** parameter of **int128** type.

For any **liquidity** < 0:

1. **SqrtPriceMath.getAmount0Delta/SqrtPriceMath.getAmount1Delta** returns **amount0/amount1** < 0 (could be zero for edge cases not dependent on liquidity)
2. **SafeCast.toUint256** reverts as the input < 0

Recommendation

We recommend replacing

```
    if (sqrtRatioX96 <= sqrtRatioAX96) {
        amount0 = SafeCast.toUint256(
            SqrtPriceMath.getAmount0Delta(
                sqrtRatioAX96,
                sqrtRatioBX96,
                liquidity
            )
        );
    } else if (sqrtRatioX96 < sqrtRatioBX96) {
        amount0 = SafeCast.toUint256(
            SqrtPriceMath.getAmount0Delta(
                sqrtRatioX96,
                sqrtRatioBX96,
                liquidity
            )
        );
        amount1 = SafeCast.toUint256(
            SqrtPriceMath.getAmount1Delta(
                sqrtRatioAX96,
                sqrtRatioX96,
                liquidity
            )
        );
    } else {
        amount1 = SafeCast.toUint256(
            SqrtPriceMath.getAmount1Delta(
                sqrtRatioAX96,
                sqrtRatioBX96,
                liquidity
            )
        );
    }
```

with

```
    if (sqrtRatioX96 <= sqrtRatioAX96) {
        int256 amount0Delta = SqrtPriceMath.getAmount0Delta(
            sqrtRatioAX96,
            sqrtRatioBX96,
            liquidity
        );
        amount0 = SafeCast.toUint256(
            amount0Delta < 0 ? -amount0Delta : amount0Delta
        );
    } else if (sqrtRatioX96 < sqrtRatioBX96) {
        int256 amount0Delta = SqrtPriceMath.getAmount0Delta(
            sqrtRatioX96,
            sqrtRatioBX96,
            liquidity
        );
        amount0 = SafeCast.toUint256(
            amount0Delta < 0 ? -amount0Delta : amount0Delta
        );
        int256 amount1Delta = SqrtPriceMath.getAmount1Delta(
            sqrtRatioAX96,
            sqrtRatioX96,
            liquidity
        );
        amount1 = SafeCast.toUint256(
            amount1Delta < 0 ? -amount1Delta : amount1Delta
        );
    } else {
        int256 amount1Delta = SqrtPriceMath.getAmount1Delta(
            sqrtRatioAX96,
            sqrtRatioBX96,
            liquidity
        );
        amount1 = SafeCast.toUint256(
            amount1Delta < 0 ? -amount1Delta : amount1Delta
        );
    }
```

Considering the current behaviour as expected one (amounts are calculated with rounding up for liquidity >= 0, otherwise - with rounding down), it is recommended to improve the **getAmountsForDelta**'s comments.

Moreover, we recommend fix the **ArrakisV2Resolver.sol:getAmountsForLiquidity** function's comments as it doesn't have the same interface as the Uniswap's function.

INFORMATIONAL-02	Inconsistent function name	Fixed at 37c595

Description

The **Underlying** library has a function **computeMintAmounts** that returns only one parameter - **mintAmount**.

Recommendation

We recommend renaming the function to **computeMintAmount**.

INFORMATIONAL-03	Zero address check	Fixed at 37c595

Description

The **ArrakisV2Storage** contracts's **_whitelistRouters** function allows to add zero address as a router.

Recommendation

We recommend adding a zero address check.

INFORMATIONAL-04	Magic number	Fixed at 37c595

Description

The **ArrakisV2Storage** contracts's **setManagerFeeBPS** function has a check for a **managerFeeBPS_** input variable, it has to be <= 10000. This number already has a named variable which could be used here.

Recommendation

We recommend replacing **10000** with **hundredPercent** from **CArrakisV2.sol**.

INFORMATIONAL-05	Zero address _arrakisV2Beacon for ArrakisV2FactoryStorage	Fixed at 37c595

Description

ArrakisV2FactoryStorage allow creating factory with zero address for **_arrakisV2Beacon**.
Also, owner can't change **arrakisV2Beacon** after creation.

Recommendation

Add zero address check for **ArrakisV2FactoryStorage** constructor

INFORMATIONAL-06	Zero address factory_ for ArrakisV2Resolver	Fixed at 37c595

Description

ArrakisV2Resolver allow creating factory with zero address for **_factory**.
Also, owner can't change **factory** after creation.

Recommendation

Add zero address check for **ArrakisV2Resolver** constructor

INFORMATIONAL-07	Inconsistent event name	Acknowledged

Description

All **ArrakisV2Storage** contract's events have a 'Log' prefix except one - **LPBurned**.

Recommendation

We recommend renaming the event to **LogLPBurned**.

Client's comments

Easier for all our indexing not to change the event names if not super necessary

STATE
MIND