



Morpheus Update - L2TokenReceiverV2 Audit Report

Version 1.0

Audited by:

HollaDieWaldfee

alexander

April 10, 2024

Contents

1	Introduction	2
1.1	About Renaissance	2
1.2	Disclaimer	2
1.3	Risk Classification	2
2	Executive Summary	3
2.1	About Morpheus Update - L2TokenReceiverV2	3
2.2	Overview	3
2.3	Issues Found	3
3	Findings Summary	4
4	Findings	5
5	Centralization Risks	6
5.1	The <code>owner</code> has to be fully trusted	6

1 Introduction

1.1 About Renaissance

Renaissance Labs was established by a team of experts including [HollaDieWaldfee](#), [MiloTruck](#), [alexander](#) and [bytes032](#).

Our founders have a distinguished history of achieving top honors in competitive audit contests, enhancing the security of leading protocols such as [Reserve Protocol](#), [Arbitrum](#), [MaiaDAO](#), [Chainlink](#), [Dodo](#), [Lens Protocol](#), Wenwin, [PartyDAO](#), [Lukso](#), [Perennial Finance](#), [Mute](#) and [Taurus](#).

We strive to deliver tailored solutions by thoroughly understanding each client's unique challenges and requirements. Our approach goes beyond addressing immediate security concerns; we are dedicated to fostering the enduring success and growth of our partners.

More of our work can be found [here](#).

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an 'as-is' and 'as-available' basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

1.3.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality
- Low - Funds are **not** at risk

1.3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

2 Executive Summary

2.1 About Morpheus Update - L2TokenReceiverV2

Morpheus introduces a new L2TokenReceiverV2 contract that now supports 2 sets of swapping parameters compared to the original L2TokenReceiver.

2.2 Overview

Project	Morpheus Update - L2TokenReceiverV2
Repository	SmartContracts
Commit Hash	753accd340ee...
Date	9 April 2024 - 10 April 2024

2.3 Issues Found

Severity	Count
High Risk	0
Medium Risk	0
Low Risk	0
Informational	1
Total Issues	1

3 Findings Summary

ID	Description	Status
I-1	Function <code>increaseLiquidityCurrentRange()</code> does not support the new swap params.	Open

4 Findings

Informational

[I-1] Function `increaseLiquidityCurrentRange()` does not support the new swap params.

Context:

- [L2TokenReceiverV2](#)

Description: In `L2TokenReceiverV2.increaseLiquidityCurrentRange()` the `token0` of the position is compared against `secondSwapParams.tokenIn`. This can be problematic in a scenario where the owner would want to add liquidity to a position that is different from the initial `wstETH / MOR`:

Assume the initial position `{tokenId:123, token0: wstETH, token1: MOR}`, where the `wstETH / MOR` pair is stored in `secondSwapParams` as part of `L2TokenReceiver__init()` and assume a position `{tokenId:456, token0: someToken, token1: MOR}`, where this would be a new pair, possibly stored in `firstSwapParams`.

Now if the owner calls `increaseLiquidityCurrentRange(tokenId:456)`, we will have `someToken != wstETH` and therefore execute the `else` branch, which will assign to `amountAdd0` the amount `rewardTokenAmountAdd_` where it should be the amount `depositTokenAmountAdd`. Similarly, `amountAdd1`, `amountMin0`, and `amountMin1` will have wrong values.

```
function increaseLiquidityCurrentRange(
    ...
) external onlyOwner returns (uint128 liquidity_, uint256 amount0_, uint256 amount1_)
{
    ...
    (, , address token0_, , , , , , ) =
    INonfungiblePositionManager(nonfungiblePositionManager).positions(
        tokenId_
    );
    if (token0_ == secondSwapParams.tokenIn) {
        ...
    } else {
        amountAdd0_ = rewardTokenAmountAdd_;
        amountAdd1_ = depositTokenAmountAdd_;
        amountMin0_ = rewardTokenAmountMin_;
        amountMin1_ = depositTokenAmountMin_;
    }
}
```

<https://github.com/MorpheusAIs/SmartContracts/blob/e4b5687dcd8d5ee323e5e97b7e420ceedbcc9c07/contracts/L2TokenReceiverV2.sol#L117-L122>

Recommendation: Depending on whether the `increaseLiquidityCurrentRange()` function is intended only for the `secondSwapParams` pair or for any pair, this issue can either be acknowledged or resolved.

5 Centralization Risks

5.1 The owner has to be fully trusted

The `L2TokenReceiverV2` contract receives the bridged yield that was generated by LPs depositing in the `Distribution` contract. The LPs have to fully trust the owner that the yield will be utilized appropriately. At any time, the owner can withdraw all of the bridged yield and all of the `UniswapV3` fees generated by the yield.

The `L2TokenReceiverV2` contract is upgradeable, and the owner can upgrade it to an arbitrary business logic. The owner has to be trusted to upgrade the contract in a non-malicious manner.