Morpheus DistributionV3 Audit Report

Version 1.0

Audited by:

HollaDieWaldfee

August 5, 2024

Contents

1 Introduction

1.1 About Renascence

Renascence Labs was established by a team of experts including HollaDieWaldfee, MiloTruck, alexxander 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 DistributionV3

`DistributionV3` makes the `lockClaim()` function accessible for private pools such that users that have staked into private pools are able to lock their claims.

2.2 Overview

Project	Morpheus DistributionV3
Contract	SmartContracts
Commit Hash	f1ff8db7c6fa...
Mitigation Hash	b0890e7dc2d4...
Date	04 August 2024 - 05 August 2024

2.3 Issues Found

Severity	**Count**
High Risk	0
Medium Risk	0
Low Risk	1
Informational	1
Total Issues	**2**

3 Findings Summary

ID	Description	Status
L-1	`_stake()` function does not check `claimLockEnd_ >= userData.claimLockEnd` for users in private pools	Resolved
I-1	Incorrect documentation for `lockClaim()` function	Resolved

4 Findings

Low Risk

[L-1] `_stake()` function does not check `claimLockEnd_ >= userData.claimLockEnd` for users in private pools

Context: DistributionV3.sol

Description: `DistributionV3._stake()` only checks `claimLockEnd_ >= userData.claimLockEnd` for users in public pools. This can lead to unintended behavior since users in private pools are now able to change their `claimLockEnd` variable. This means the `owner` must always query the latest `claimLockEnd` for any user it wants to stake for to not accidentally set `claimLockEnd` to a lower value.

It is possible to perform the `claimLockEnd_ >= userData.claimLockEnd` for public and private pools. The downside is that the user would be able to set a high `claimLockEnd` and the owner couldn't set it back to a lower value. However, by design a user in a private pool should now be able to set any `claimLockEnd`, after all that's the point of making `lockClaim()` accessible. As a result, the downside is negligible and does not have a security impact.

Recommendation: Consider checking `claimLockEnd` regardless of whether the pool is private.

```
## DistributionV3.sol

          if (claimLockEnd_ == 0) {
              claimLockEnd_ = userData.claimLockEnd > block.timestamp ?
              userData.claimLockEnd : uint128(block.timestamp);
          }
+         require(claimLockEnd_ >= userData.claimLockEnd, "DS: invalid claim lock
end");

          if (pool.isPublic) {
              require(amount_ > 0, "DS: nothing to stake");
-             require(claimLockEnd_ >= userData.claimLockEnd, "DS: invalid claim lock
end");
```

Morpheus: Fixed.

Renascence: The recommendation has been implemented.

Informational

[I-1] Incorrect documentation for `lockClaim()` function

Context: IDistributionV3.sol

Description: The documentation for the `lockClaim()` function in `IDistributionV3` is wrong since the function is not used to withdraw tokens. Instead, it is used to lock rewards.

```
## IDistributionV3.sol

-     * The function to withdraw tokens from the pool.
+     * The function to lock rewards.
```

Morpheus: Fixed.

Renascence: The recommendation has been implemented.