Morpheus DistributionV2 Audit Report

Version 1.0

Audited by:

HollaDieWaldfee

alexxander

July 25, 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 DistributionV2

DistributionV2 is an upgrade for the Distribution contract which introduces claim locks. By locking their rewards, users can multiply them. The multiplier mirrors the dilution rate the user experiences and decreases over time. A detailed specification of the change can be found in MRC42.

2.2 Overview

Project	Morpheus DistributionV2
Repository	SmartContracts
Commit Hash	59a65f8adbf9...
Mitigation Hash	e4c41547e50d...
Date	19 July 2024 - 22 July 2024

2.3 Issues Found

Severity	**Count**
High Risk	2
Medium Risk	0
Low Risk	1
Informational	1
Total Issues	**4**

3 Findings Summary

ID	Description	Status
H-1	`lockClaim()` function allows to decrease `claimLockEnd` and earn inflated multiplier	Resolved
H-2	`totalVirtualDeposited` accounting breaks after the upgrade and pool reward rate gets diluted	Resolved
L-1	`lockClaim()` resets `lastStake` to `block.timestamp` even though no stake has occurred	Resolved
I-1	Improvements	Resolved

4 Findings

High Risk

[H-1] `lockClaim()` function allows to decrease `claimLockEnd` and earn inflated multiplier

Context: DistributionV2.sol

Description: The `lockClaim()` function fails to check that `claimLockEnd_ > userData.claimLockEnd`. It only checks that `claimLockEnd_ > block.timestamp`. This allows an attacker to first set `claimLockEnd_` to a value such that they earn the maximum multiplier. By leveraging `lockClaim()`, they can decrease `claimLockEnd_` at any time, being able to earn the inflated multiplier without actually locking the earned rewards.

Recommendation: In `lockClaim()`, add a `claimLockEnd_ > userData.claimLockEnd` check.

Morpheus: Fixed.

Renascence: The recommendation has been implemented.

[H-2] `totalVirtualDeposited` accounting breaks after the upgrade and pool reward rate gets diluted

Context: DistributionV2.sol

Description: The upgrade to `DistributionV2` introduces three new fields to the `UserData` struct:

```
## IDistributionV2.sol

struct UserData {
    uint128 lastStake;
    uint256 deposited;
    uint256 rate;
    uint256 pendingRewards;
    // Storage changes for DistributionV2
    uint128 claimLockStart;
    uint128 claimLockEnd;
    uint256 virtualDeposited;
}
```

Before the upgrade to V2, the `deposited` variable has been used instead of `virtualDeposited`. It is not possible to simply replace `deposited` with `virtualDeposited` due to the specific case when `deposited > 0` but `virtualDeposited = 0`.

There are five instances of this issue, but it can be illustrated based on one instance. Here is the instance in the `claim()` function:

```
## DistributionV2.sol

poolData.totalVirtualDeposited =
    poolData.totalVirtualDeposited +
    userData.deposited -
    userData.virtualDeposited;
```

In the case where `virtualDeposited = 0`, the `totalVirtualDeposited` variable is updated incorrectly. The calculation subtracts zero because `virtualDeposited` has not been set yet. This case must be checked for and `deposited` must be used instead of `virtualDeposited`.

Recommendation: If `virtualDeposited = 0`, the `deposited` variable should be used instead of `virtualDeposited`.

```
## DistributionV2.sol

          // Update pool data
          poolData.lastUpdate = uint128(block.timestamp);
          poolData.rate = currentPoolRate_;
+         uint256 deposited = userData.virtualDeposited == 0 ? userData.deposited :
userData.virtualDeposited;
          poolData.totalVirtualDeposited =
              poolData.totalVirtualDeposited +
              userData.deposited -
-             userData.virtualDeposited;
+             deposited;

          // Update user data
          userData.rate = currentPoolRate_;
@@ -220,7 +221,8 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable
          // Update pool data
          poolData.lastUpdate = uint128(block.timestamp);
          poolData.rate = currentPoolRate_;
-         poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - userData.virtualDeposited;
+         uint256 deposited = userData.virtualDeposited == 0 ? userData.deposited :
userData.virtualDeposited;
+         poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - deposited;

          // Update user data
          userData.lastStake = uint128(block.timestamp);
@@ -283,7 +285,8 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable
          // Update pool data
          poolData.lastUpdate = uint128(block.timestamp);
          poolData.rate = currentPoolRate_;
-         poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - userData.virtualDeposited;
+         uint256 deposited = userData.virtualDeposited == 0 ? userData.deposited :
userData.virtualDeposited;
+         poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - deposited;

          // Update user data
          userData.lastStake = uint128(block.timestamp);
@@ -339,7 +342,8 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable
          // Update pool data
          poolData.lastUpdate = uint128(block.timestamp);
          poolData.rate = currentPoolRate_;
-         poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - userData.virtualDeposited;
```

```
+        uint256 deposited = userData.virtualDeposited == 0 ? userData.deposited :
userData.virtualDeposited;
+        poolData.totalVirtualDeposited = poolData.totalVirtualDeposited +
virtualDeposited_ - deposited;

         // Update user data
         userData.rate = currentPoolRate_;
@@ -358,7 +362,8 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable
     }

     function _getCurrentUserReward(uint256 currentPoolRate_, UserData memory
     userData_) private pure returns (uint256) {
-        uint256 newRewards_ = ((currentPoolRate_ - userData_.rate) *
userData_.virtualDeposited) / PRECISION;
+        uint256 deposited = userData_.virtualDeposited == 0 ? userData_.deposited :
userData_.virtualDeposited;
+        uint256 newRewards_ = ((currentPoolRate_ - userData_.rate) * deposited) /
PRECISION;

         return userData_.pendingRewards + newRewards_;
     }
```

Morpheus: Fixed here and here.

Renascence: The recommendation has been implemented. The cases where `userData.virtualDeposited == 0` are accounted for and `userData.deposited` is used instead.

Low Risk

[L-1] `lockClaim()` resets `lastStake` to `block.timestamp` even though no stake has occurred

Context: DistributionV2.sol

Description: The `lockClaim()` function sets `userData.lastStake = block.timestamp`, even though `lockClaim()` does not update the amount of staked funds, it just extends to claim lock end. As a result, after calling `lockClaim()`, a user needs to wait `withdrawLockPeriodAfterStake` seconds to be able to withdraw funds.

Recommendation: Consider not setting `userData.lastStake` in `lockClaim()`.

Morpheus: Fixed.

Renascence: The recommendation has been implemented.

Informational

[I-1] Improvements

Context: DistributionV2.sol

Description: Rename `depoisted_` to `deposited_`.

```
## DistributionV2.sol

          userData.pendingRewards = _getCurrentUserReward(currentPoolRate_, userData);

-         uint256 depoisted_ = userData.deposited + amount_;
+         uint256 deposited_ = userData.deposited + amount_;
          uint256 multiplier_ = _getClaimLockPeriodMultiplier(uint128(block.timestamp),
          claimLockEnd_);
-         uint256 virtualDeposited_ = (depoisted_ * multiplier_) / PRECISION;
+         uint256 virtualDeposited_ = (deposited_ * multiplier_) / PRECISION;

          // Update pool data
          poolData.lastUpdate = uint128(block.timestamp);
@@ -288,7 +288,7 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable
          // Update user data
          userData.lastStake = uint128(block.timestamp);
          userData.rate = currentPoolRate_;
-         userData.deposited = depoisted_;
+         userData.deposited = deposited_;
          userData.virtualDeposited = virtualDeposited_;
          userData.claimLockStart = uint128(block.timestamp);
          userData.claimLockEnd = claimLockEnd_;
```

Rename `multipier` to `multiplier`.

```
## DistributionV2.sol

     function _getClaimLockPeriodMultiplier(uint128 start_, uint128 end_) internal
     pure returns (uint256) {
         uint256 powerMax = 16_613_275_460_000_000_000; // 16.61327546 * DECIMAL

-        uint256 maximalMultipier_ = 10_700_000_000_000_000_000; // 10.7 * DECIMAL
-        uint256 minimalMultipier_ = DECIMAL; // 1 * DECIMAL
+        uint256 maximalMultiplier_ = 10_700_000_000_000_000_000; // 10.7 * DECIMAL
+        uint256 minimalMultiplier_ = DECIMAL; // 1 * DECIMAL

         uint128 periodStart_ = 1721908800; // Thu, 25 Jul 2024 12:00:00 UTC
         uint128 periodEnd_ = 2211192000; // Thu, 26 Jan 2040 12:00:00 UTC TODO
@@ -434,12 +434,12 @@ contract DistributionV2 is IDistributionV2, OwnableUpgradeable,
UUPSUpgradeable

         uint256 endPower_ = _tanh(2 * (((end_ - periodStart_) * DECIMAL) /
         distributionPeriod));
         uint256 startPower_ = _tanh(2 * (((start_ - periodStart_) * DECIMAL) /
         distributionPeriod));
-        uint256 multipier_ = (powerMax * (endPower_ - startPower_)) / DECIMAL;
+        uint256 multiplier_ = (powerMax * (endPower_ - startPower_)) / DECIMAL;

-        multipier_ = multipier_ > maximalMultipier_ ? maximalMultipier_ : multipier_;
-        multipier_ = multipier_ < minimalMultipier_ ? minimalMultipier_ : multipier_;
+        multiplier_ = multiplier_ > maximalMultiplier_ ? maximalMultiplier_ :
multiplier_;
+        multiplier_ = multiplier_ < minimalMultiplier_ ? minimalMultiplier_ :
multiplier_;

-        return (multipier_ * PRECISION) / DECIMAL;
+        return (multiplier_ * PRECISION) / DECIMAL;
     }
```

Remove redundant assignment of `userData.claimLockEnd`.

```
## DistributionV2.sol

         userData.deposited = newDeposited_;
         userData.virtualDeposited = virtualDeposited_;
         userData.claimLockStart = uint128(block.timestamp);
-        userData.claimLockEnd = userData.claimLockEnd;

         if (pool.isPublic) {
             totalDepositedInPublicPools -= amount_;
```

Morpheus: Fixed here and here.

Renascence: The recommendation has been implemented.