Rebalance Contracts Audit

Oleksandr Zadorozhnyi
Viacheslav Vozniuk

May, 2024

Email: info@4irelabs.com

Website: 4irelabs.com

CONFIDENTIAL

This document and the Code Review it references is strictly private, confidential and personal to its recipients and should not be disclosed, copied, distributed, or reproduced in whole or in part, not passed to any third party.

1. **Introduction.** Rebalance requested 4irelabs a review of their contracts.

   There are 13 contracts source code in scope (first review on commit **e0147c1f55e8e12d280b96bcca89e1ae22ae7a7d**):

   - ProtocolAccessControl.sol
   - InterestVaultV1.sol
   - VaultRebalancerV1.sol
   - VaultPermit.sol
   - RebalancerManager.sol
   - InterestLocker.sol
   - ProviderManager.sol
   - AaveV3Sepolia.sol
   - CompoundV3Arbitrum.sol
   - RadiantV2Arbitrum.sol
   - AaveV3Arbitrum.sol
   - LodestarArbitrum.sol
   - DForceArbitrum.sol

2. **Warranty.** Audit is provided on an "as is" basis, without warranty of any kind, express or implied. The Auditor does not guarantee that the Code Review will identify all instances of security vulnerabilities or other related issues.

3. **Executive Summary.** Provided contracts realize managed vault logic on top of other lending markets. Users can deposit and withdraw their assets to the vaults which are managed by the address with the Rebalancer role. Admins can grant Rebalancer roles and change the most important protocol parameters, including fees, treasury address, minimum deposit amount, etc. There are no known compiler bugs, for the specified compiler versions, that might affect the logic of the contracts.

4. **Critical Bugs and Vulnerabilities.** No Critical and Major issues have been found.

5. LINE BY LINE REVIEW.

5.1 VaultRebalancer.sol:

5.1.1 *Note*: Line 41. Constructor's NatSpec missing description for *userDepositLimit_*, *vaultDepositLimit_* and *withdrawFeePercent_* parameters.

5.1.2 *Note*: Line 61. Redundant sanity checks in the constructor, same validations would be performed in the '_setDepositLimits' function.

5.1.3 *Low*: Line 128. Vault contact includes the '_checkRebalanceFee()' function to check if rebalancer provides a reasonable fee amount that is less than max fee percent, however fee charging could be executed through rebalance multiple times in a row, forcing users to pay unreasonable fees that they not expected.
Recommendation: Consider adding a restriction mechanism based for example on a time lock pattern that would disallow charging fees multiple times in a short period.

5.2 InterestVaultV1.sol:

5.2.1 *Low*: Line 121. If the 'initializeVaultShares()' function would not be called during vault deployment, this would open room for the execution of the inflation attack.
Recommendation: Consider adding a check to the deposit and mint function that the vault is initialized or make sure that vault initialization is executed at the same time as vault deployment.

5.2.2 *Low*: Line 167. The 'approve' function saves *allowance* value in terms of underlying assets while users call it with a *shares* amount. Over time the exchange rate between shares and assets would grow, meaning that previously set allowance could be not enough to fully utilize the user's *shares* balance, this would break users expectations.
Recommendation: Consider saving allowance value within *shares* instead of converting it to *assets*.

5.2.3 *Low*: Line 679. The 'setTreasury()' function fails to check if the provided address is not zero. Setting *treasury* to zero address accidentally or intentionally could potentially DOS withdrawing and rebalancing processes for tokens that revert on transfer to the zero address.
Recommendation: Consider adding zero address checks for the *treasury* address in the 'setTreasury()' function and constructor.

5.3 RebalancerManager.sol:

5.3.1 *Low*: Line 28, 64. The *allowedExecutor* mapping and the 'allowExecutor' function provide access control and duplicate the ProtocolAccessControl.sol functionality.
Recommendation: Add the Executor role and use ProtocolAccessControl.sol for access control.

5.3.2 *Low*: Line 48, 90. When rebalancing a vault with all provider funds (assets == type(uint256).max), the provider method 'getDepositBalance' is called twice with the same parameters.
Recommendation: Call the 'getDepositBalance' provider method once and continue working with the balance value. If the '_checkAssetsAmount' method is not planned to be used elsewhere, move the provider's funds check to the 'rebalanceVault' method and remove the '_checkAssetsAmount' method.

5.4 InterestLocker.sol:

5.4.1 *Low*: Line 91. The *TokensLocked* event is missing the *unlockTime* parameter from the created lock.
Recommendation: Add the *unlockTime* parameter to the *TokensLocked* event.

5.4.2 *Note*: Line 102, 104. Before reading the *LockInfo* data, it is necessary to check whether the beneficiary is the sender.
Recommendation: Move sender validation before reading the *LockInfo* data.

5.4.3 *Note*: Line 37. Missing NatSpec comments for all functions.
Recommendation: Add all NatSpec comments.

5.5 AaveV3Sepolia.sol, AaveV3Arbitrum.sol:

5.5.1 *Note*: Line 21, 32, 46, 55. Different names for the same logical variable.
Recommendation: Use one variable name.

5.5.2 *Note*: Line 21, 22, 32, 33, 46, 47, 55, 56. Variables are created that are used once.
Recommendation: Use '_getPool' instead of the created variable.

5.5.3 *Note*: Line 37. Missing NatSpec comments for the '_getPool' function.
Recommendation: Add a comment to the '_getPool' function.

5.6 RadiantV2Arbitrum.sol:

5.6.1 *Note*: Line 21, 31, 48, 59. Different names for the same logical variable.
Recommendation: Use one variable name.

5.6.2 *Note*: Line 21, 22, 31, 32, 48, 49, 59, 60. Variables are created that are used once.
Recommendation: Use '_getPool' instead of the created variable.

5.7 LodestarArbitrum.sol:

5.7.1 *Low*: Line 22. Import "hardhat/console.sol".
Recommendation: Remove import.