Mellow protocol: Gearbox integration Security Analysis

by Pessimistic

This report is public

December 06, 2022

Abstract

In this report, we consider the security of smart contracts of
Mellow protocol: Gearbox integration project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

Summary

In this report, we considered the security of Mellow protocol: Gearbox integration smart contracts. We described the audit process in the section below.

The audit showed several issues of low severity. After the additional testing Mellow protocol team discovered and fixed a single critical issue and two issues of medium severity. We have included the information about these issues into the report.

The project meets high code quality standards.

After the initial audit the codebase was updated. Most of the issues have been fixed. Three low severity issues were commented. One new low severity issue has been found.

General recommendations

We recommend fixing the remaining issue.

Project overview

Project description

For the audit, we were provided with Mellow protocol: Gearbox integration project on a public GitHub repository, commit ec21a37fd67206abf9cc636156fb1f7cb4e68631.

The scope of the audit included only the following files:

- **contracts/interfaces/vaults/IGearboxRootVault.sol**
- **contracts/interfaces/vaults/IGearboxVault.sol**
- **contracts/interfaces/vaults/IGearboxVaultGovernance.sol**
- **contracts/vaults/GearboxRootVault.sol**
- **contracts/vaults/GearboxVault.sol**
- **contracts/vaults/GearboxVaultGovernance.sol**
- **contracts/vaults/IntegrationVault.sol**
- **contracts/utils/GearboxHelper.sol**
- dependencies

The documentation for the project included **Gearbox_Vault_docs.pdf** file, sha1sum `6346f7b8bfef09efaba935d98455c69ae3a76950`.

132 tests out of 132 for the foundry and 1447 tests out of 1450 for hardhat were successfully passed. The code coverage (calculated in foundry) for the audit volume is 70.08%. Disabled tests are considered failed.

The total LOC of audited sources is 1623.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit e5033e1b2190c60f65823fb1a0f68d27aa1accd1. This update includes fixes for most of the issues mentioned in the initial audit and an increase in the number of tests. These changes also contain fixes for all issues discovered by the project team.

141 tests out of 141 for the foundry and 1471 tests out of 1475 for hardhat were successfully passed. The code coverage (calculated in foundry) for the audit volume is 75.50%. Disabled tests are considered failed.

Audit process

We started the audit on November 7 and finished on November 23, 2022.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. After a discussion, we performed preliminary research of the protocols that the project integrates with:

- Gearbox,
- Curve,
- Convex.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- The logic of withdrawal is sound and safe;
- The slippage protection is implemented properly;
- The protocol fee is charged correctly and securely.

We scanned the project with the static analyzer Slither with our own set of rules and then manually verified all the occurrences found by the tool.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we received a new commit with the updated codebase. This update contains fixes for some issues from our report and new tests. We have also added issues to this report that were discovered and fixed by the developers.

We have checked the fixes for all the issues in this report. We also found one new issue.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

C01. **Multiple withdrawals of one deposit** (fixed)

In the **GearboxRootVault** contract, the user calls the `registerWithdrawal` method. The LP tokens are still in his wallet. Then this user sends his LP tokens to another address and again makes a `registerWithdrawal` from another address. Then everything repeats for the following address. Thus, the user can withdraw all tokens from the contract balance.

The issue has been fixed and is not present in the latest version of the code.

Medium severity issues

Medium issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. **Incorrect calculations for tvl** (fixed)

In the **GearboxRootVault** contract, an incorrect calculation is made for the withdrawal of assets in the `invokeExecution` method. It is necessary to call the `closeCreditAccount` method in the **GearboxVault**. Only after that it is possible to get the tvl value from the **GearboxVault**.

The issue has been fixed and is not present in the latest version of the code.

M02. **Incorrect parameter** (fixed)

In the **IGearboxVaultGovernance** contract, the `univ3Adapter` parameter is common to all `GearboxVaults`. But `GearboxVaults` can be opened for different `CreditManagers`. These parameters should be different for each `CreditManager`.

The issue has been fixed and is not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. CEI pattern violation (commented)

In the **GearboxRootVault** contract, the `withdraw` function calls the `_pull` function, which performs an external call at line 140 of the **AggregateVault** contract. However, the `withdraw` function updates state variables after the call and thus violates the CEI pattern. In this case, the function performs an external call to vault token contracts and, therefore, can be considered safe. Nevertheless, we highly recommend following CEI pattern to verify the predictability of the code execution.

> *Comment from the developers:* *The `call` from the `subvault` is actually part of the withdrawal in our system (since the `subvault` can be our own or approved by us volts), accordingly, we believe that there is no CEI violation here, since all interaction takes place inside our system.*

L02. Gas consumption (fixed)

In the `deposit` function of the **GearboxRootVault** contract, consider moving the `_chargeFees` call at line 151 to the `else` block at lines 158–162 since this function charges fees only in the case of non-zero supply.

The issue has been fixed and is not present in the latest version of the code.

L03. Redundant check (fixed)

In `_chargePerformanceFees` function of the **GearboxRootVault** contract, the check for `baseSupply == 0` at line 404 is redundant since the contract calls this function only from the `_chargeFees` function, which verifies that `baseSupply > 0` due to the condition at lines 349–351.

The issue has been fixed and is not present in the latest version of the code.

L04. Redundant function (commented)

The internal function `_setProtocolParams` of the **VaultGovernance** contract is unused.

> *Comment from the developers:* *The function can be used in the future when adding other vaults to our system.*

L05. Unchecked returned values (commented)

In the **GearboxRootVault** contract, `addDepositorsToAllowlist` and `removeDepositorsFromAllowlist` functions allow adding and removing depositors to the allowlist using `add` and `remove` methods of `AddressSet` struct. These methods of the struct return a value depending on the result. However, these values are never checked in the **GearboxRootVault** contract.

> *Comment from the developers: The `add` function returns a value depending on whether the element was previously in the set. This fact is insignificant to us and does not require verification.*

L06. Missing events (fixed)

The **GearboxRootVault** contract does not emit any events when adding or removing depositors to/from the allowlist. We recommend emitting events when changing important parameters to improve system maintainability.

The issue has been fixed and is not present in the latest version of the code.

L07. Incompliance with OZ library rules (commented)

According to OZ documentation, the overriding hook functions should follow two rules of hooks:

1. It should have the `virtual` attribute.
2. It should always call the parent's hook in the beginning.

However, the `_beforeTokenTransfer` function does not follow any of these rules in the **GearboxRootVault** contract.

> *Comment from the developers: In our case, we do not need the function to be virtual (since no child contracts are planned by the Protocol), calling super is also not much relevant, since the function in the parent contract doesn't run any code. Hence, we think it is not critical here not to follow this recommendation.*

Notes

N01. Uncontrolled transfer of approved tokens (commented)

In the **IntegrationVault** contract, once a user approves tokens for the contract address, anyone can transfer these tokens within the approved amount from this user to the contract using the `transferAndPush` function with arbitrary from address. In this case, the owner of the tokens will not receive any LP tokens in return.

> *Comment from the developers: The user should not give approval for **IntegrationVault** type contracts. All interaction with the system goes through **AggregateVault** contracts, they are given approvals.*

N02. Outdated EVM version (addressed)

The project uses an outdated `evmVersion` `istanbul`.

This analysis was performed by Pessimistic:

Vladimir Tarasov, Security Engineer
Vladimir Pomogalov, Security Engineer
Yhtyyar Sahatov, Junior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

December 06, 2022