Mean Finance Security Analysis

by Pessimistic

This report is public

November 21, 2021

Abstract

In this report, we consider the security of smarts contracts of Mean Finance project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of Mean Finance smart contracts. We performed our audit according to the procedure described belows.

The initial audit showed two issues of medium severity: Bug and Use of deprecated function. Also, several low-severity issues were found.

After the initial audit, the code base was updated. All the issues were fixed, new tests were added to to the project. Also, the developers revealed a bug in **ChainlinkOracle** contract and fixed it. However, after the update, one test fails to pass.

General recommendations

We recommend adding CI to run tests, calculate code coverage, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with Mean Finance project on a private GitHub repository, commit c5d9dc07d081add1293c6e82372c2f3b621ec79c.

The project has a documentation at https://docs.mean.finance/ and as a separate file **Mean Finance v1.1.pdf**, sha1sum is 344e4ce72fb821100333a176a478896f2b940558. All the interfaces have detailed NatSpec comments.

All 635 tests pass, the code coverage is 98.98%.

The total LOC of audited sources is 1800.

Code base update

After the initial audit, the code base was updated. For the recheck, we were proviede with commit 950e7cd578f9b29016aa860c37ae407e59500ca1.

In this update, all the mentioned issues were fixed.

Also, new tests were added to the project, the overall code coverage was increased to 98.99%. However, one test out of 648 does not pass.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
    - We scan project's code base with automated tool SmartCheck.
    - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
    - We manually analyze code base for security vulnerabilities.
    - We assess overall project structure and quality.
- Report
    - We reflect all the gathered information in the report.

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 often lead to the loss of funds or other catastrophic failures. The contracts should not be deployed before these issues are fixed. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in 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.

Bug (fixed)

When dealing with tightly packed flags in `toUInt8()` function of **PermissionMath** contract, `|=` operator is the preferred way to set a bit. Unlike `+=` operator, it correctly processes attempts to raise the flag multiple times.

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

Use of deprecated function (fixed)

Function `latestAnswer` is deprecated and is not recommended to use. Or If it cannot be replaced, consider adding checks for returned values at lines 229 and 249 of **ChainlinkOracle** contract.

For details, read OpenZeppelin blog.

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

Tests issue (new)

After the code base update, one test out of 648 fails to pass in **ChainlinkOracle** contract.

Low severity issues

Low severity issues do not directly affect project's operations. However, they might lead to various problems in the future versions of the code. We recommend taking them into account.

Code quality (fixed)

- Consider declaring functions as `external` instead of `public` when possible to improve code readability and optimize gas consumption:
  - Function `FEE_PRECISION()` of **DCAHubConfigHandler** contract, line 93.
  - Function `swap()` of **DCAHubSwapHandler** contract, line 200.

  The issues have been fixed and are not present in the latest version of the code.

- In **DCAHubPositionHandler** contract, CEI pattern is violated at line 63.

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

- Consider splitting the project into separate modules to make the logic more transparent. Swap data should not be modified in **DCAHubPositionHandler** contract.

  > *Comment from developers: initially we did just that, we only modified the swap amount delta. But after some tests, we realized that it was cheaper to add the next swap's amounts to the SwapData. We got 50k gas savings with that change.*

  With the comment from developers, we consider this entry as not an issue.

Notes

Token decimals limit

Function `_getDecimals()` of **ChainlinkOracle** contract will not work if provided token has more than 128 decimals.

This analysis was performed by Pessimistic:
Daria Korepanova, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

November 21, 2021