Nereus Protocol Contracts Code Audit by Ambisafe Inc.

March, 2022

Oleksii Matiiasevych

1. **Introduction.** Nereus Finance. requested Ambisafe to perform a code audit of the contracts implementing Nereus Protocol. The contracts in question can be identified by the following git commit hash:

   `bbf2bde432c23bf0ee535c4bdccde12f5a1130cd`

   The scope of the audit is the modifications of Aave Protocol V2 and an additional 5 contracts located in the staking folder.

   After the initial code audit, Nereus Finance team applied a number of updates which can be identified by the following git commit hash:

   `5329684cc602c16eca6a8816dfaaa2462c31097d`

   Additional verification was performed after that.

2. **Disclaimer.** The code audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts for any specific purpose, or their bugfree status. The code audit documentation below is for internal management discussion purposes only and should not be used or relied upon by external parties without the express written consent of Ambisafe.

3. **Executive Summary.** All the initially identified, minor and above, severity issues were **fixed** and are not present in the **final version** of the contracts. There are **no** known compiler bugs for the specified compiler version (0.6.12), that might affect the contracts' logic. There were 0 critical, 1 major, 1 minor, 23 informational and optimizational issues identified in the initial version of the contracts. The non-informational issues found in the contract were not present in the final version. They are described below for historical purposes.
   Modifications of the Aave Protocol V2 mainly consist of introducing a liquidator role,

making the **liquidationCall()** function only accessible to certain addresses, and splitting the protocol profits between the treasury and fee distributor. Changes also allow emergency admin to perform pool configuration actions.

4. **Critical Bugs and Vulnerabilities.** No critical bugs or vulnerabilities were found.
5. **Initial Line By Line Review.**

    5.1. UiPoolDataProviderV2, line 220. Note, the **bytes32ToString** function **for** loop condition could be optimized to '**i < bytesArray.length**'.

    5.2. LendingPool, line 89. Note, in the expression '**liquidators[msg.sender] == true**' there is no point in comparing a boolean value to **true**, just use the boolean value itself.

    5.3. LendingPool, line 94. Note, the **LendingPoolConfigurator** role is set to the **LendingPoolConfigurator** contract by default. That contract in turn is controlled by the **LendingPoolAdmin** role, but lacks functionality that would call liquidators management functions in the **LendingPool** contract. Consider changing the management access rights to the **LendingPoolAdmin** role directly, or adding the necessary functions to the **LendingPoolConfigurator** contract.

    5.4. ChefIncentivesController, line 37. Optimization, the poolConfigurator variable could be made immutable.

    5.5. ChefIncentivesController, line 39. Optimization, the rewardMinter variable could be made immutable.

    5.6. ChefIncentivesController, line 56. Typo, **poitns** should be points.

    5.7. ChefIncentivesController, line 58. Note, misleading comment about startTime variable. It is not a block number but a timestamp.

    5.8. MasterChef, line 36. Optimization, the poolConfigurator variable could be made immutable.

    5.9. MasterChef, line 38. Optimization, the rewardMinter variable could be made immutable.

    5.10. MasterChef, line 55. Typo, **poitns** should be points.

    5.11. MasterChef, line 57. Note, misleading comment about startTime variable. It is not a block number but a timestamp.

    5.12. MerkleDistributor, line 23. Note, instead of '86400 * 365' consider using '365 days' for better readability.

5.13. MerkleDistributor, line 24. Note, instead of '86400 * 7' consider using '7 days' for better readability.

5.14. MerkleDistributor, line 26. Optimization, the **rewardMinter** could be made immutable.

5.15. MultiFeeDistribution, line 51. Optimization, the **treasury** variable could be made immutable.

5.16. MultiFeeDistribution, line 59. Note, instead of '86400 * 7' consider using '7 days' for better readability.

5.17. MultiFeeDistribution, line 162. Optimization, use **rewards[i].token** instead of **rewardTokens[i]** to avoid excessive storage reads.

5.18. MultiFeeDistribution, line 175. Minor, the **unlockedBalance** function should use **user** parameter instead of **msg.sender**.

5.19. MultiFeeDistribution, line 294. Major, **_updateReward(user)** should be executed in the beginning of the **mint()** function, otherwise the user will get extra rewards.

5.20. MultiFeeDistribution, line 384. Note, instead of '86400' consider using '1 day' for better readability.

5.21. MultiFeeDistribution, line 509. Note, the **RewardAdded** event is not used.

5.22. MultiFeeDistribution, line 513. Note, the **RewardsDurationUpdated** event is not used.

5.23. TokenVesting, line 10. Note, instead of '86400 * 365' consider using '365 days' for better readability.

5.24. TokenVesting, line 13. Optimization, the **minter** variable could be made immutable.

5.25. TokenVesting, line 14. Optimization, the **owner** variable could be made immutable.

Oleksii Matiiasevych