Ribbon Autocall Smart Contracts Review

By: ChainSafe Systems

July 2023

Ribbon Autocall Smart Contracts Review

Auditors: Anderson Lee, Tanya Bushenyova, Oleksii Matiiasevych

WARRANTY

Introduction

Ribbon Finance requested ChainSafe Systems to perform a review of the contracts implementing their Autocall vault. The contracts can be identified by the following git commit hash:

`9a7c788f123cf1e82b207b1ddbcddcab14727019`

The scope included RibbonAutocallVault.sol and the difference between RibbonTreasuryVault.sol and RibbonTreasuryVaultLite.sol.

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

`01c716a6bf452bdf8789f8cf5b3ef3e36db2da8e`

Additional verification was performed after that.

Disclaimer

The review makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about the fitness of the contracts for any specific purpose, or their bug free status.

Executive Summary

There are no known compiler bugs for the specified compiler version (0.8.4), that might affect the contracts' logic.

There were 0 critical, 0 major, 1 minor, 17 informational/optimizational issues identified in the initial version of the contracts. All the issues found in the contracts were not present in the final verified version of the contracts. They are described below for historical purposes. We enjoyed working with the Ribbon Finance team, and liked how engaged they were in the discussion and improvement process throughout the review.

Critical Bugs and Vulnerabilities

No critical issues were identified.

Line by Line Review. Fixed Issues

1. RibbonAutocallVault.sol, line 4: Note, `SafeMath` import is not used.

2. RibbonAutocallVault.sol, line 6: Note, `SafeERC20` import is not used.

3. RibbonAutocallVault.sol, line 116: Optimization, the `initialize()` function reads `period` from storage multiple times, consider storing it in a local variable.

4. RibbonAutocallVault.sol, line 202: Optimization, the `setPeriodAndObservationFrequency()` function reads `period` from storage multiple times, consider storing it in a local variable.

5. RibbonAutocallVault.sol, line 206: Note, the `setPeriodAndObservationFrequency()` function only checks that `_period > 0`. During initialization the `VaultLifecycleTreasury.verifyInitializerParams()` function performs more checks of the `_period` value.

6. RibbonAutocallVault.sol, line 207: Minor, in the `setPeriodAndObservationFrequency()` function `period` should be replaced with `_period` in the `_obsFreq` check. Next observation frequency should evenly divide the next period.

7. RibbonAutocallVault.sol, line 358: Note, the `_setPutOptionPayoff()` function name is confusing: it implies a setter while it's a view function (getter).

8. RibbonAutocallVault.sol, line 376: Optimization, in the `_setPutOptionPayoff()` function there's no need to do the payoff calculations if `_nOptionType != OptionType.DIP`, it could be just assigned 0.

9. RibbonAutocallVault.sol, line 400: Optimization, the `_couponsEarned()` function reads `reserveRatio` from storage multiple times, consider storing it in a local variable.

10. RibbonAutocallVault.sol, line 421: Optimization, the `_couponsEarned()` function reads `couponState.couponType` from storage multiple times, consider storing it in a local variable.

11. RibbonAutocallVault.sol, line 450: Optimization, the `_autocallState()` function reads `obsFreq` from storage multiple times, consider storing it in a local variable.

12. RibbonAutocallVault.sol, line 496: Optimization, the `_lastObservation()` function reads `numTotalObs` from storage multiple times, consider storing it in a local variable.

13. RibbonAutocallVault.sol, line 499: Optimization, the `_lastObservation()` function reads `obsFreq` from storage multiple times, consider storing it in a local variable.

14. RibbonTreasuryVault.sol, line 38: Note, no need to use `SafeMath` for the 0.8.x Solidity version because `SafeMath` is default in the 0.8.x Solidity version.

15. RibbonTreasuryVault.sol, line 152: Note, `InitiateGnosisAuction` event is not used.

16. RibbonTreasuryVault.sol, line 455: Note, in the `_removeDepositor()` function there is no need to update `array[arrayLength - 1]` with `array[i]` because `array[arrayLength - 1]` is deleted when `array.pop()` is performed.

17. RibbonTreasuryVault.sol, line 458: Note, in the `_removeDepositor()` function once `excludeDepositor` is deleted from the array, no need to iterate over the array anymore. A break from the for loop can be performed.

18. RibbonTreasuryVaultLite.sol, line 70: Note, no need to use `SafeMath` for the 0.8.x Solidity version because `SafeMath` is default in the 0.8.x Solidity version.

Anderson Lee

Tanya Bushenyova

Oleksii Matiiasevych