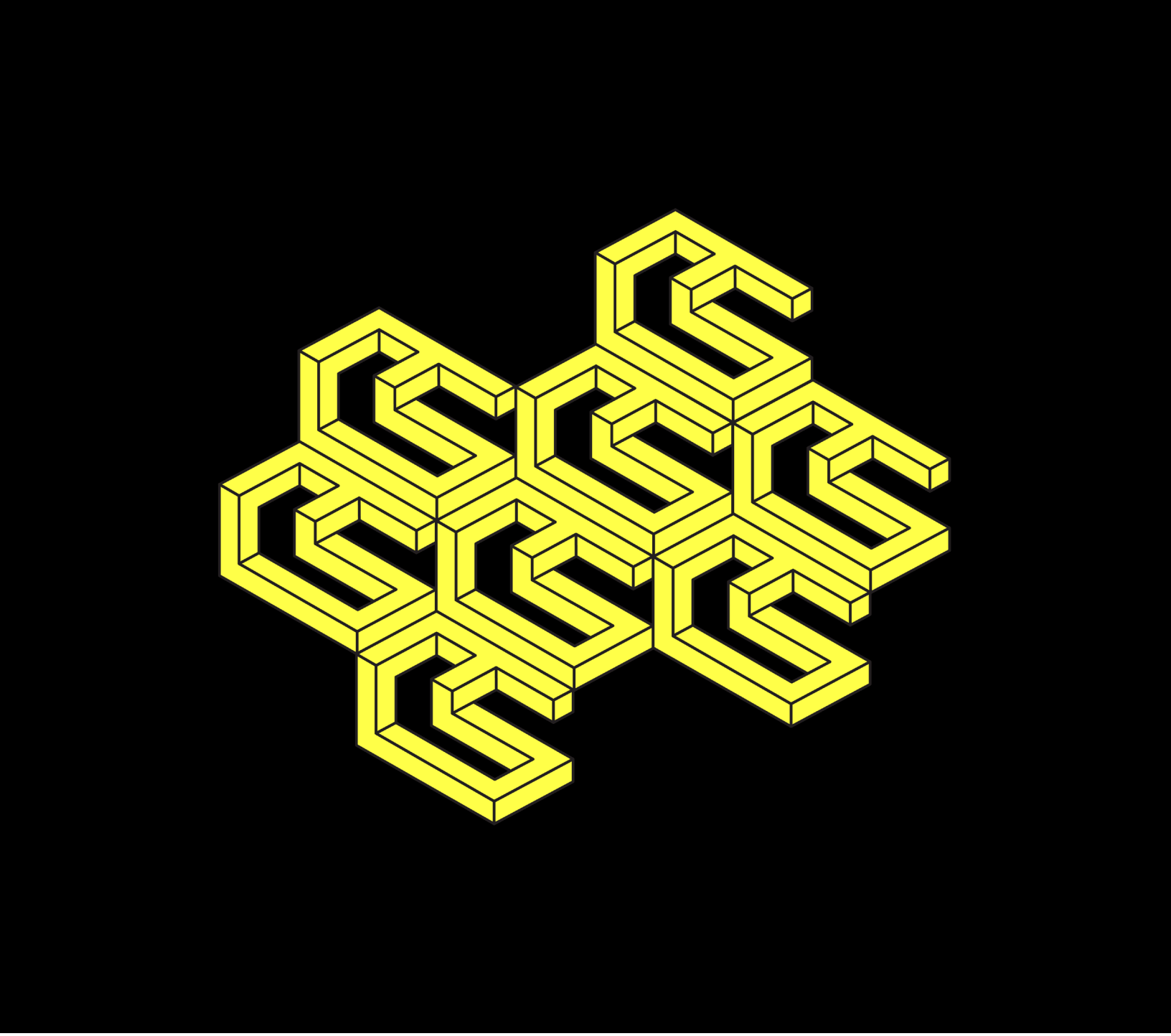

RibbonFinance Smart Contracts Review

By ChainSafe Systems

April 2021

ChainSafe

Ribbon Finance Smart Contracts Review

Auditors: Tanya Bushenyova, Oleksii Matiiasevych

Warranty

Executive Summary

There were 1 critical, 3 major, 0 minor, 17 informational/optimizational issues identified in this version of the contracts. There are **no** known compiler bugs, for the specified compiler version (0.7.2), that might affect the contracts’ logic.

Update Verification Summary

There were 0 critical, 0 major, 0 minor, 3 informational/optimizational issues identified in this version of the contracts. There are **no** known compiler bugs, for the specified compiler version (0.7.2), that might affect the contracts’ logic.

1. Introduction

Ribbon Finance requested ChainSafe Systems to perform a review of the RibbonCoveredCall and parts of GammaAdapter and ProtocolAdapter smart contracts. The contracts in question can be identified by the following git commit hash:

`d1102fbd7c59e14e371b71b27cf6accb92d997fb`

There are 3 contracts/libraries/interfaces in scope.

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

`269e56845e76849855abe20f2dea5c5c5da52750`

Additional verification was performed after that, results of which are included in the Appendix A.

2. Disclaimer

The review 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 bug free status.

Since this is a partial audit, our report does not guarantee a lack of vulnerabilities in the remainder of the Ribbon codebase. Our reported results will resolve as follows: This action will be performed, but the result of it is out of scope of this audit.

3. Executive Summary

There were 1 critical, 3 major, 0 minor, 17 informational/optimizational issues identified in this version of the contracts. There are **no** known compiler bugs, for the specified compiler version (0.7.2), that might affect the contracts' logic.

RibbonCoveredCall manages deposited funds to create options on the OPYN platform. This review is made on **assumption** that OPYN platform is secure and safe to use. If that assumption is wrong, then all deposited funds could be lost.

RibbonCoveredCall by itself is deployed using an **upgradable proxy** pattern that is supposed to be managed by Ribbon multisig wallet. This implies that the following review only applies to the exact version identified by the commits in section (1). As soon as Ribbon will decide to utilize the upgrade function, this review becomes void, as the upgrade could change any piece of logic, **ultimately taking hold of any deposited funds.**

It is users' responsibility to make sure that Ribbon multisig is controlled by a sufficient number of trustful members.

We believe that while the upgradability could be useful, and safe, on early stages of the protocol, the increase of value locked in the contract might cause unnecessary psychological pressure on the `manager` (being a group might lower the risk, but not remove it), by creating an incentive to seize the power for themself and take hold of all the funds.

We enjoyed working with the Ribbon Finance team, they were eager to discuss all the feedback and swift in applying necessary changes.

4. Critical Bugs and Vulnerabilities

One critical issue was identified during the review. It could allow some users to withdraw almost double of their share of deposits in case of `emergencyWithdrawFromShort` use. More details in 5.14.

5. Line-by-line review

5.1. RibbonCoveredCall, line 28. Note, the `factory` variable is not used.

5.2. RibbonCoveredCall, line 73. Note, the comments section is outdated.

5.3. RibbonCoveredCall, line 92. Note, 0 decimals could be valid for some tokens.

5.4. RibbonCoveredCall, line 111. Note, the comments section is outdated.

5.5. RibbonCoveredCall, line 168. **Major**, the `withdrawalFee` could be set to almost 100% effectively blocking any withdrawals. Consider setting some reasonable limit to the maximum percentage.

5.6. RibbonCoveredCall, line 216. Note, the revert message could be imprecise due to a `MINIMUM_SUPPLY` being set in the constructor.

5.7. RibbonCoveredCall, line 219. Optimization, `totalWithDepositedAmount >= MINIMUM_SUPPLY` check could be moved closer to the beginning of the function to revert earlier, saving gas.

5.8. RibbonCoveredCall, line 220. Note, the revert message could be imprecise due to a `MINIMUM_SUPPLY` being set in the constructor.

5.9. RibbonCoveredCall, line 270. **Major**, as the current version of the contract expects to work only with `Call` options, consider adding a validation of the `optionTerms` type. Otherwise there could be an inconsistency between what is set as the `nextOption` and what will actually be used.

5.10. RibbonCoveredCall, line 292. Note, the `rollToNextOption` function could be called multiple times for the same `nextOption`. Consider restricting this by clearing the `nextOption` variable.

5.11. RibbonCoveredCall, line 296. Optimization, `block.timestamp > nextOptionReadyAt` check could be moved closer to the beginning of the function to revert earlier, saving gas.

5.12. RibbonCoveredCall, line 301. **Major**, in case of a compromised `manager` or too high demand for withdrawals. it would make sense to have separate functions for `closeShort` and `createShort` and to have `closeShort` be callable by anyone when the option is already expired. It would unlock the funds for withdrawal before locking them in the next option. In the current version there is no delay between options to let users withdraw all the funds.

5.13. RibbonCoveredCall, line 314. Optimization, `otoken.collateralAsset()` could be replaced by the `asset` variable as the current version of the contract only works with options where underlying and collateral are the same.

5.14. RibbonCoveredCall, line 332. **Critical**, during the `emergencyWithdrawFromShort` function call, all the tokens get back to the `RibbonCoveredCall` contract, but the `lockedAmount` remains unchanged (is not set to zero). Now the first holders to withdraw will get ~double of their share of tokens because of the `total = lockedAmount.add(currentAssetBalance);` will be incorrect.

5.15. RibbonCoveredCall, line 358. Note, an offchain check needs to be performed for each created option to make sure that the whole `oToken` balance is finally swapped (sold). Otherwise if some `oTokens` are left unsold, then after the `exercise` some of the profits will be left stuck.

5.16. RibbonCoveredCall, line 409. Note, the revert message could be imprecise due to a `MINIMUM_SUPPLY` being set in the constructor.

5.17. RibbonCoveredCall, line 413. Note, the revert message could be imprecise due to a `MINIMUM_SUPPLY` being set in the constructor.

5.18. RibbonCoveredCall, line 452. Optimization, `lockedAmount.add(assetBalance())` could be changed to `lockedAmount.add(withdrawableBalance)` in order to save gas.

5.19. ProtocolAdapter, line 191. Style, the `revertWhenFail` helper could be utilized here.

5.20. IProtocolAdapter, line 16. Note, invalid comment. USDC is the `strikeAsset`.

5.21. OptionVaultStorage, line 48. Note, invalid comment. 0.005 means 0.5% fee.

Tanya Bushenyova

Oleksii Matiiasevych

Appendix A: Smart Contracts Update Verification

Auditors: Tanya Bushenyova, Oleksii Matiiasevych

1. Executive Summary

There were 0 critical, 0 major, 0 minor, 3 informational/optimizational issues identified in this version of the contracts. There are no known compiler bugs, for the specified compiler version (0.7.2), that might affect the contracts' logic.

RibbonCoveredCall manages deposited funds to create options on the OPYN platform. This review is made on **assumption** that OPYN platform is secure and safe to use. If that assumption is wrong, then all deposited funds could be lost.

RibbonCoveredCall by itself is deployed using an **upgradable proxy** pattern that is supposed to be managed by Ribbon multisig wallet. This implies that the following review only applies to the exact version identified by the commits in section (1). As soon as Ribbon will decide to utilize the upgrade function, this review becomes void, as the upgrade could change any piece of logic, **ultimately taking hold of any deposited funds**.

It is **users' responsibility** to make sure that Ribbon multisig is controlled by a **sufficient number of trustful members.**

We believe that while the upgradability could be useful, and safe, on early stages of the protocol, the increase of value locked in the contract might cause unnecessary psychological pressure on the `manager` (being a group might lower the risk, but not remove it), by creating an incentive to seize the power for themself and take hold of all the funds.

We enjoyed working with the Ribbon Finance team, they were eager to discuss all the feedback and swift in applying necessary changes.

All the significant issues were addressed in this update, and 0 new issues were found.

2. Remaining Findings

2.1. RibbonCoveredCall, line 97. Note, 0 decimals could be valid for some tokens.

2.2. RibbonCoveredCall, line 333. Note, if the `rollToNextOption` will not be called before the `nextOption` expiration date, then the contract will be stuck in the current state. All the users will have to withdraw their deposits and move on to the next version. Consider adding an additional separate step to reset the next option.

2.3. RibbonCoveredCall, line 400. Note, an offchain check needs to be performed for each created option to make sure that the whole `oToken` balance is finally swapped (sold). Otherwise if some `oTokens` are left unsold, then after the `exercise` some of the profits will be left stuck.