ReSource Network Contracts Code Audit by Ambisafe Inc.
March, 2022
Yaroslav Tumanov

CONFIDENTIAL

This document and the Code Audit it references is strictly private, confidential and personal to its recipients and should not be disclosed, copied, distributed or reproduced in whole or in part, not passed to any third party.

YOU MAY NOT ISSUE ANY PRESS RELEASE ABOUT THIS CODE AUDIT. AMBISAFE IS PROVIDING THIS CODE AUDIT AS PART OF OUR QUALITY ASSURANCE PROCESS ON A “BEST EFFORTS”, “AS IS” AND “AS AVAILABLE” BASIS AT THIS PARTICULAR POINT AND TIME, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

WITHOUT LIMITING THE FOREGOING, AMBISAFE EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, TITLE, ACCURACY AND COMPLETENESS, OR FUNCTIONING OF THIS CODE.

BY ACCESSING THIS DOCUMENT YOU ACKNOWLEDGE, ACCEPT AND AGREE TO THE FOREGOING. THIS DOCUMENT IS NOT FOR PUBLICATION OR ANY DISTRIBUTION, DIRECTLY OR INDIRECTLY, IN WHOLE OR IN PART. THESE MATERIALS ARE NOT AN OFFER OF SECURITIES FOR SALE. SECURITIES MAY NOT BE OFFERED OR SOLD IN THE UNITED STATES IN ABSENCE OF REGISTRATION WITH THE U.S. SECURITIES AND EXCHANGE COMMISSION OR AN EXEMPTION FROM REGISTRATION UNDER

THE U.S. SECURITIES ACT OF 1933, AS AMENDED.

1. **Introduction.** ReSource requested Ambisafe to perform a code audit of the contracts implementing the ReSource Network product. The source code with contracts in question can be identified by the following github links:
https://github.com/ReSource-Network/ReSource-Protocol/tree/master/packages/hardhat/contracts/Credit
https://github.com/ReSource-Network/ReSource-Protocol/tree/master/packages/hardhat/contracts/Network

   and git commit hash:

   `87cca59e4cd1e7ac1354b12d99d1363d76a57693`

   There are 11 contracts/libraries in scope.

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

   `32fe2d06ee365dbe677426ab0f8f271b2ba9a5f0`

   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 contract except for 1 minor issue. Previously 1 critical issue was found, described in 5.11 and fixed by the ReSource team. There are **no** known compiler bugs for the specified compiler version (0.8.0), that might affect the contracts' logic. There were 1 critical, 6 major, 7 minor, 33 informational issues identified in the initial version of the contracts.

4. **Critical Bugs and Vulnerabilities.** 1 Critical issue was found and described in 5.11 and fixed.

5. Initial Line By Line Review.

5.1. CreditFeeManager, line 66. Major, change to **+=creditFee** to not override accruedFees each time.

5.2. CreditFeeManager, line 70. Optimization, **distributeFees** function visibility should be changed to **external**.

5.3. CreditFeeManager, line 102. Note, could be removed: **_feePercent >= 0.**

5.4. CreditFeeManager, line 185. Note, modifier **onlyUnderwriter** is not used and could be removed.

5.5. CreditManager, line 56. Major, it is possible to set createCreditLine more than once for the same network/networkMember. It will increase the **increaseTotalCredit** twice and rewrite values inside **setCreditLimit**. Possible solution: do not allow to override credit Line.

5.6. CreditManager, line 77. Minor, the function **extendCreditLine** should have **onlyRegisteredNetwork(_network**) modifier.

5.7. CreditManager, line 90. Minor, the function **swapCreditLinePool** should have **onlyRegisteredNetwork(_network)** modifier.

5.8. CreditManager, line 92. Major, **swapCreditLinePool** function should decreaseTotalCredit in the old pool and increase in the new pool.

5.9. CreditManager, line 97. Optimization, make **creditLine** memory. Reading from memory is cheaper.

5.10. CreditManager, line 116. Major, **creditLine.creditPool** always == address(0) at this point because it is deleted on line 114.

5.11. CreditManager, line 165. **Critical**, wrong comparison. It should be **creditLine.issueDate + creditLineExpiration < block.timestamp**. Consider using helper time functions **passed(timestamp) notPassed(timestamp)**. Details:

5.11.1. **createCreditLine** called and new credit line with current time created.

5.11.2. Calling **closeCreditLine** will be successful next 180 days and will fail after 180 days.

5.11.3. Recommendation: **closeCreditLine** should become callable 180 days after created credit line and not earlier.

5.12. CreditManager, line 220. Minor, make the divide operation last to not lose cents.

5.13. CreditManager, line 256. Note, change CreditRequest to CreditManager.

5.14. CreditManager, line 262. Note, change CreditRequest to CreditManager.

5.15. CreditPool, line 28. Optimization, rewardsDuration, periodFinish, lastUpdateTime variables could fit in one storage slot (uint32,uint32,uint32).

5.16. CreditPool, line 69. Note, revert reason could be helpful here.

5.17. CreditPool, line 77. Note, the **viewMapping** function could be removed, because **rewardData** is public.

5.18. CreditPool, line 134. Optimization, safeMath is not needed in solidity > 0.8.x and could be removed.

5.19. CreditPool, line 140. Major, add the **onlyCreditFeeManager** modifier to disallow calls from others. Right now anyone can force anyone else to stake all their balance (if there are enough allowance).

5.20. CreditPool, line 189. Note, change CollateralPool to CreditPool. Revert reason is empty.

5.21. CreditRequest, line 62. Minor, it is possible to approve an empty request. Possible solution is to add
require(requests[_network][_networkMember].creditLimit != 0, "CreditRequest: Request not exists");

5.22. CreditRequest, line 81. Optimization, make **request** memory. Reading from memory is cheaper.

5.23. CreditRequest, line 93. Note, if **request.creditLimit <= curCreditLimit** then nothing will happen and the request will be deleted.

5.24. CreditRequest, line 153. Note, the **getCreditRequest** function could be removed, because **requests** are public.

5.25. CreditRoles, line 38. Minor, the function **revokeOperator** should have **operatorExist(_operator)** modifier.

5.26. PriceOracle, line 13. Major, if this is the Production version, then **onlyOwner** or other admin modifier should be added here.

5.27. ICreditPool, line 9. Note, use prefix _ for param names everywhere like in other places.

5.28. ICreditRequest, line 8. Optimization, **creditLimit** could be stored as uint128 if it fits to use one storage cell for bool,bool,uint128, check possible max creditLimit.

5.29. ICreditRequest, line 15. Optimization, **creditLimit** could be stored as uint128 if it fits to use one storage cell for bool,uint128, check possible max creditLimit.

5.30. ICreditRequest, line 22. Optimization, **creditLimit** could be stored as uint128 if it fits to use one storage cell for bool,uint128, check possible max creditLimit.

5.31. CIP36, line 34. Optimization, creditBalanceOf function visibility could be moved from public to external.

5.32. CIP36, line 50. Optimization, the **setCreditLimit** function visibility could be moved from public to external.

5.33. NetworkFeeManager, line 49. Minor, the **_totalFeePercent** variable should be checked.

5.34. NetworkFeeManager, line 54 Note, wrong revert reason, 'Ambassador fee percent greater than 100' should be used instead.

5.35. NetworkFeeManager, line 103. Minor, **distributeFees** function will emit an event with **AmbassadorRewardsUpdated(address(0), 0)** in case ambassador is zero address. Looks like a**ddress(this)** should be used.

5.36. NetworkFeeManager, line 105. Optimization, **uint256 networkFee = totalFees - ambassadorFee** could be used.

5.37. NetworkFeeManager, line 112. Optimization, **updateAmbassadorFeePercent** function visibility could be moved from public to external.

5.38. NetworkFeeManager, line 115. Note, wrong revert reason, 'Ambassador fee percent greater than 100' should be used instead.

5.39. NetworkRoles, line 37. Note, a revert message will be helpful.

5.40. NetworkRoles, line 93. Note, no need for **notNull** modifier here, because in case of zero address it will fail in **ambassadorExists.**

5.41. NetworkRoles, line 154. Note, the NetworkSet event could be helpful here.

5.42. NetworkRoles, line 177. Note, modifier **memberDoesNotExist** is not used and could be removed.

5.43. NetworkRoles, line 187. Note, modifier **onlyMember** is not used and could be removed.

5.44. NetworkRoles, line 215. Note, modifier **operatorExists** is not used and could be removed.

5.45. RUSD, line 49. Note, modifier **onlyNetworkOperator** is not used and could be removed.

5.46. RUSD, line 89. Optimization, the **bulkTransfer** function visibility could be moved from public to external.

5.47. RUSD, line 90. Note, revert reason could be helpful.

6. Line By Line Verification. Remaining and Acknowledged Issues.

6.1. CreditRoles, line 38. Minor, the function revokeOperator should have operatorExist(_operator) modifier.

6.2. CIP36, line 34. Optimization, **creditBalanceOf** function visibility could be moved from public to external.

6.3. CIP36, line 50. Optimization, the **setCreditLimit** function visibility could be moved from public to external.

6.4. NetworkRoles, line 154. Note, the NetworkSet event could be helpful in the **setNetwork** function.