Rezerve

Smart Contract Security Audit

Date 9/August/2021

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and ApeAudits and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (ApeAudits) owe no duty of care towards you or any other person, nor does ApeAudits make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and ApeAudits hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, ApeAudits hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against ApeAudits, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Purpose

This document details ApeAudits' findings and recommended solutions. Our audit was performed in the weeks leading up to Rezerve's launch. The contracts we audited were Rezerve.sol, RezerveExchange.sol, and RezerveStakingReceiver.sol

Findings

NO.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	N/A	Passed
2	Race Conditions Audit	N/A	Passed
3	Authority Control Audit	Permission Vulnerability Audit Excessive Auditing Authority	Passed Passed
4	Safe Design Audit	Zeppelin Module Safe Compiler Version Hard-coded Version Fallback Function Safeuse Show Coding Security Function Return Value Security Call Function Security	Passed Passed Passed Passed Passed Passed Passed
5	Denial of Service Audit	N/A	Passed
6	Gas Optimization Audit	N/A	Passed
7	Design Logic Audit	N/A	Passed
8	Malicious Event Log Audit	N/A	Passed
9	"False Deposit" Vulnerability Audit	N/A	Passed
10	Uninitialized Storage Pointers Audit	N/A	Passed
11	Arithmetic Accuracy Deviation Audit	N/A	Passed

Issue List

NO.	Issue Description	Issue Severity	Issue Status
1	Public functions that should be private	High	Fixed
2	Improper calculations in swap function	High	Fixed
3	Incorrect sender in swap function	High	Fixed
4	Heavily centralized owner permissions	High	To be fixed
5	Improper calculation in transfer function	High	Fixed
6	Excessive exclusion from rewards	High	Fixed
7	Ownership can be reclaimed	Moderate	Fixed
8	Improper exclusion of owner	Moderate	Fixed
9	Improper calculation of tax deductions	Moderate	Fixed
10	Sell logic improperly implemented	Moderate	Fixed
11	Redundant blacklist functionality	Moderate	Fixed
12	Numerical errors in exchange calculation	Moderate	Fixed
13	Redundant transfers in exchange	Moderate	Fixed
14	Inefficient frontend for staking	Moderate	Fixed
15	DAI holders unable to sell	Moderate	Intentional
16	Redundant SafeMath	Low	Fixed

High Severity Issues

- **Issue #1:**
  Public functions `swapIt(uint256)` and `swapTokensForDai(uint256)` should be declared private to avoid malicious usage.

  Comments:
  Fixed in revision 2 by making both functions internal.

- **Issue #2:**
  In `swapIt(uint256)`, the exchange/staking shares are improperly computed to be much larger than the input, which could prevent transfers after the contract reaches the swap threshold.

  Comments:
  Fixed in revision 2.

- **Issue #3:**
  Calling `transfer(address, uint256)` inside `swapIt(uint256)` causes the contract to attempt to transfer from the message sender instead of the contract's stored balance. This causes the contract to attempt to blacklist the sender, causing the parent transfer to fail. This will happen once the contract is over the swap threshold and `stakingTax` is true.

  Comments:
  Fixed in revision 3 by calling `_tokenTransfer(address, address, uint256, bool)` instead.

- **Issue #4:**
  The owner account has extremely liberal permissions, including withdrawing all LP tokens owned by the contract.

  Comments:
  The Rezerve team has assured us that these functions are to be used in good faith, and that they intend to slightly decentralize these permissions by transferring ownership to a multisig wallet.

- **Issue #5:**
  While migrating from SafeMath to regular checked math, a `.add()` was changed to a minus, causing transactions to fail.

  Comments:
  Fixed in revision 2.

- **Issue #6:**
  As a feature of the contract, addresses are excluded from reflection rewards. However, all transfers depend on the list of excluded addresses, and excluding new addresses increases the length of the `excluded` array. This causes the gas cost for transfers to increase with each new holder of the token, and causes all transfers to become excessively expensive very quickly.

  Comments:
  Fixed in revision 1 by removing the line that was excluding holders from rewards.

Moderate Severity Issues

- **Issue #7:**
  Ownership of the contract can be reclaimed after renouncing if the contract is first locked and unlocked.

  Comments:
  Fixed in revision 2 by resetting the previous owner when unlocking.

- **Issue #8:**
  In the contract's constructor, the owner is excluded from rewards without calling `excludeFromReward(address)`. This will cause the owner's token holdings to always be uncounted in the contract's supply calculations.

  Comments:
  Fixed in revision 2 by calling `excludeFromReward(address)`.

- **Issue #9:**
  In `_getRValues(uint256, uint256, uint256, uint256)`, the variable `rTransferAmount` is computed without considering `tLiquiditySale` from `_getTValues(uint256)`, causing the ratio of r-values to t-values to not be preserved.

  Comments:
  Fixed in revision 2 by computing `rLiquiditySale` and subtracting it from `rTransferAmount`.

- **Issue #10:**
  In `_transfer(address, address, uint256)`, the contract implements sell-specific logic by checking if the receiver address is the Uniswap router instead of the Uniswap pair. This causes the sell logic to be skipped as the router is never the receiver.

  Comments:
  Fixed in revision 2 by correctly checking the receiver.

- **Issue #11:**
  In `_transfer(address, address, uint256)`, the contract blacklists addresses that send tokens twice in the same block, but it also reverts transfers that occur within 20 seconds of each other. This causes blacklisting to never happen.

  Comments:
  Fixed in revision 2 by only reverting transfers if the address is not blacklisted. This allows the blacklist to be updated and subsequent transfers for the blacklisted address will fail.

- **Issue #12:**
  In `exchangeAmount(uint256)` in RezerveExchange, the correct amount is computed, but it does so with the possibility of numerical errors because of large multiplication and division. The amount can be computed more accurately by simply computing the amount * DAI balance / current supply.

  Comments:
  Fixed in revision 3.

- **Issue #13:**
  In `exchangeReserve(uint256)` in RezerveExchange, two transfers are performed where only one is needed. This could cause taxes to be unintentionally applied twice.

  Comments:
  Fixed in revision 3 by transferring directly to the burn address.

- **Issue #14:**
  In RezerveStaking, two variables of type `mapping(uint256 => address)` are used to associate a holder's address with their staking vault. This will force frontends to either store holder addresses and their vaults in a database or iterate through all staking vaults until the right one is found. This is much less efficient than simply using a `mapping(address => address)`.

  Comments:
  Fixed in revision 2 by using a much simpler contract.

- **Issue #15:**
  If token holders also hold DAI, they are unable to sell.

  Comments:
  The Rezerve team has told us that this is a feature of the contract and will be communicated clearly to token holders.

Low Severity Issues

- **Issue #16:**
  The contract uses an old version of SafeMath that is no longer needed after Solidity version 0.8.0.

  Comments:
  Fixed in revision 2 by removing SafeMath.

Conclusion

All issues fixed, except owner privileges. However, the Rezerve team has assured us that they intend to act in good faith and will also protect the owner account with a multisig wallet.

Appendix

All supporting files can be found at https://github.com/ApeAudits1/Rezerve-extras