FungieDAO

Smart Contract Security Audit

Date 7/September/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 September 7, 2021. The contract we audited is located at https://bscscan.com/address/0x8918Bb91882CE41D9D9892246E4B56e4571a9fd5.

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	Incorrect destination of swap tokens	High	Fixed
2	Numerical error	Moderate	Fixed
3	Incorrect calculations	Moderate	Fixed
4	Public functions that should be onlyOwner	Moderate	Fixed
5	Excessive permission in Child function	Low	Will be fixed in future Child contracts
6	Use of arrays instead of mappings	Low	Not fixed

High Severity Issues

- **Issue #1:**
  In `swap_bnb_for_tokens(uint256)`, the swapped tokens are sent to `msg.sender` instead of the address of the contract. This causes liquidity to be added using the contract's existing token balance, which is usually 0 and will cause the transaction to fail. Also, Uniswap/Pancakeswap pairs will block the receiver from being the contract's address.

  Comments:
  Fixed in revision 3 by sending the tokens to a burn address, setting the burn address balance to 0, and then adding the burned amount to the contract's balance.

Moderate Severity Issues

- **Issue #2:**
  In `distributeTax_child()`, the variable `percent_parent_convertible` is computed in a roundabout way that introduces numerical error. It can be hardcoded or computed directly from the parent and child tax values.

  Comments:
  Fixed in revision 4 by hardcoding a 46% split, which is roughly 3/6.5.

- **Issue #3:**
  In `distributeTax_child()`, the variable `parent_tax_amount` is incorrectly computed. It should be divided by the total tax, not by 100.

  Comments:
  Fixed in revision 4.

- **Issue #4:**
  The functions `distributeTax_child()`, and `add_child_bnb_to_liquidity()` are declared public with no restrictions. These functions will likely only be called by the owners, and leaving the functions open to be called by other contracts will add attack surface to the system.

  Comments:
  Fixed in revision 4 by restricting `add_child_bnb_to_liquidity()` to only the owner contract and `distributeTax_child()`to either the owner or parent contract. However, there is no code in the parent contract capable of calling this function.

Low Severity Issues

- **Issue #5:**
  The `ownerOrParent` modifier for `distributeTax_child()`introduced in revision 4 allows the child contract deployer to incorrectly specify the parent contract and then distribute collected funds to that address instead of to the parent.

  Comments:
  This issue was raised after the FungieDAO team had already deployed the contract, and so it cannot be changed in the deployed code. However, the currently deployed code does not use the child contract code at all, so this issue is only a code cleanliness issue. The FungieDAO team has assured us that this issue will be fixed for all child contracts that will be deployed.

- **Issue #6:**
  Arrays are used for tracking blacklisted addresses and child contracts, where mappings would be much more efficient.

  Comments:
  Not fixed.

Conclusion

All high and medium severity issues fixed. There is an issue with the Child contract code as it is currently written in the deployed contract that would allow the Child deployer to steal collected funds. However, this code is not used in the current contract and was mainly included for auditing, and the FungieDAO team has told us that they will fix this issue in all child contracts. One more low severity issue causes blacklist and child tracking functionality to be less gas efficient, but this is not currently an issue because Binance Smart Chain gas fees are very low.

Appendix

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