BLOCK AUDIT REPORT

Smart Contract Security Audit Report

Block Audit Report Team received the TunaDog team's application for smart contract security audit of the TunaDog Token on July 03, 2021. The following are the details and results of this smart contract security audit:

Token Name: TunaDog

The Contract address: 0xc69B1BA0CD292a85d80180FfFFD8746A915a26FF

Link Address: https://bscscan.com/address/0xc69B1BA0CD292a85d80180FfFFD8746A915a26FF

The audit items and results:

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

Audit Result: Passed

Audit Number: BAR005103072021

Audit Date: July 03, 2021

Audit Team: Block Audit Report Team

Table of Content

Introduction

This Audit Report mainly focuses on the overall security of TunaDog Smart Contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Block Audit Report team has performed rigorous testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.

In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analyzing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

Audit Details

Project Name: TunaDog
Website/ Bscscan Code (Mainnet):
0xc69B1BA0CD292a85d80180FfFFD8746A915a26FF
Languages: Solidity (Smart contract)

Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Mythril, Contract Library, Slither

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security

Identifying security related issues within each contract and the system of contract.

Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Issue Categories

Every issue in this report was assigned a severity level from the following:

High level severity issues

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium level severity issues

1. Issues on this level could potentially bring problems and should eventually be fixed.

Low level severity issues

Issues on this level are minor details and warningJs that can remain unfixed but would be better fixed at some point in the future.

Issues Checking Status

№	Issue description.	Checking status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Functions Outline

+ interface IERC20
  - totalSupply()
  - balanceOf(address account)
  - transfer(address recipient, ...
  - allowance(address owner, add ...
  - approve(address spender, uin ...
  - transferFrom(address sender, ...
+ library SafeMath
  - tryAdd(uint256 a, uint256 b)
  - trySub(uint256 a, uint256 b)
  - tryMul(uint256 a, uint256 b)
  - tryDiv(uint256 a, uint256 b)
  - tryMod(uint256 a, uint256 b)
  - add(uint256 a, uint256 b)
  - sub(uint256 a, uint256 b)
  - mul(uint256 a, uint256 b)
  - div(uint256 a, uint256 b)
  - mod(uint256 a, uint256 b)
  - sub(uint256 a, uint256 b, st ...
  - div(uint256 a, uint256 b, st ...
  - mod(uint256 a, uint256 b, st ...
  - msgSender()
  - msgData()
+ library Address
  - isContract(address account)
  - sendValue(address payable re ...
  - functionCall(address target, ...
  - functionCall(address target, ...
  - functionCallWithValue(addres ...
  - functionCallWithValue(addres ...
  - functionStaticCall(address t ...
  - functionStaticCall(address t ...
  - functionDelegateCall(address ...
  - functionDelegateCall(address ...
  - verifyCallResult(bool succe ...
  - owner()
  - renounceOwnership()

- transferOwnership(address ne ...

+ - interface IUniswapV2Factory
  - feeTo()
  - feeToSetter()
  - getPair(address tokenA, addr ...
  - allPairs(uint)
  - allPairsLength()
  - createPair(address tokenA, a ...
  - setFeeTo(address)
  - setFeeToSetter(address)

+ interface IUniswapV2Pair
  - name()
  - symbol()
  - decimals()
  - totalSupply()
  - balanceOf(address owner)
  - allowance(address owner, add ...
  - approve(address spender, uin ...
  - transfer(address to, uint va ...
  - transferFrom(address from, a ...
  - DOMAINSEPARATOR()
  - PERMITTYPEHASH()
  - nonces(address owner)
  - permit(address owner, addres ...
  - MINIMUMLIQUIDITY()
  - factory()
  - token0()
  - token1()
  - getReserves()
  - price0CumulativeLast()
  - price1CumulativeLast()
  - kLast()
  - burn(address to)
  - swap(uint amount0Out, uint a ...
  - skim(address to)
  - sync()
  - initialize(address, address)

+ interface IUniswapV2Router01
  - factory()
  - WETH()
  - addLiquidity(
  - addLiquidityETH(
  - removeLiquidity(

- removeLiquidityETH(
- removeLiquidityWithPermit(
- removeLiquidityETHWithPermit ...
- swapExactTokensForTokens(
- swapTokensForExactTokens(
- swapExactETHForTokens(uint a ...
- swapTokensForExactETH(uint a ...
- swapExactTokensForETH(uint a ...
- swapETHForExactTokens(uint a ...
- quote(uint amountA, uint res ...
- getAmountOut(uint amountIn, ...
- getAmountIn(uint amountOut, ...
- getAmountsOut(uint amountIn, ...
- getAmountsIn(uint amountOut, ...

+ interface IUniswapV2Router02 is IUnis ...
  - removeLiquidityETHSupporting ...
  - removeLiquidityETHWithPermit ...
  - swapExactTokensForTokensSupp ...
  - swapExactETHForTokensSupport ...
  - swapExactTokensForETHSupport ...
+ contract TunaDog is Context, IERC2 ... *
  - name()
  - symbol()
  - decimals()
  - totalSupply()
  - balanceOf(address account)
  - transfer(address recipient, ...
  - allowance(address owner, add ...
  - approve(address spender, uin ...
  - transferFrom(address sender, ...
  - increaseAllowance(address sp ...
  - decreaseAllowance(address sp ...
  - isExcludedFromReward(address ...
  - totalFees()
  - deliver(uint256 tAmount)
  - reflectionFromToken(uint256 ...
  - tokenFromReflection(uint256 ...
  - setDeadAddress(address payab ...
  - setMarketingAddress(address ...
  - getmarketingWallet()
  - getdeadWallet()
  - setshouldSwapToBNB(bool enab ...
  - excludeFromReward(address ac ...

- includeInReward(address acco ...
- transferBothExcluded(addres ...
- burn(uint256 value)
- burn(address who, uint256 ...
- excludeFromFee(address accou ...
- includeInFee(address account ...
- setTaxFeePercent(uint256 tax ...
- setMarketingFeePercent(uint2 ...
- setDeadFeePercent(uint256 de ...
- setLiquidityFeePercent(uint2 ...
- setMaxTxPercent(uint256 maxT ...
- setSwapAndLiquifyEnabled(boo ...
- reflectFee(uint256 rFee, ui ...
- getValues(uint256 tAmount)
- getTValues(uint256 tAmount)
- getRValues(TData memory d ...
- getRate()
- getCurrentSupply()
- takeLiquidity(uint256 tLiqu ...
- addAddress(address adr)
- takeMarketing(uint256 tMark ...
- takeDead(uint256 tDead)
- calculateTaxFee(uint256 amo ...
- calculateMarketingFee(uint25 ...
- calculateDeadFee(uint256 am ...
- calculateLiquidityFee(uint25 ...
- removeAllFee()
- restoreAllFee()
- isExcludedFromFee(address ac ...
- approve(address owner, addr ...
- transfer(
- swapAndLiquify(uint256 contr ...
- swapTokensForEth(uint256 tok ...
- addLiquidity(uint256 tokenAm ...
- tokenTransfer(address sende ...
- transferStandard(address se ...
- transferToExcluded(address ...
- transferFromExcluded(addres ...

Manual Audit:

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM and Kovan networks to test the contract functionality.

Critical Severity Issues

No critical severity issues found.

High Severity Issues

No high severity issues found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

No low severity issues found.

Owner privileges

No

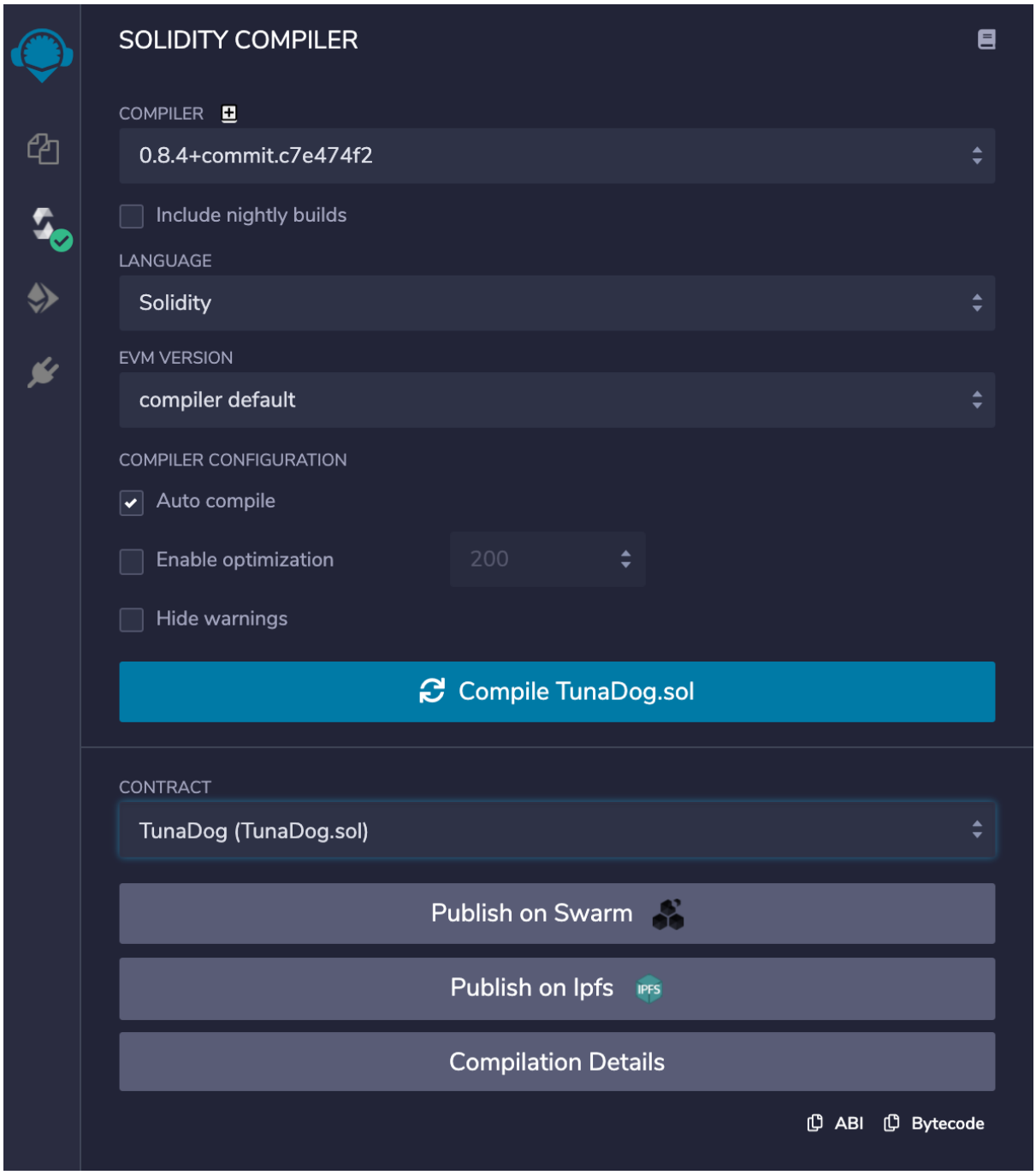
Ownership Renounced

Yes

Automated Audit

Remix Compiler Warnings

It throws warnings by Solidity's compiler. If it encounters any errors the contract cannot be compiled and deployed.



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 below 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 BlockAudit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (BlockAudit) owe no duty of care towards you or any other person, nor does BlockAudit 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 BlockAudit 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, BlockAudit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against BlockAudit, 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.

Summary

Smart contracts do not contain any high severity issues!

Note:
Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

BLOCK AUDIT REPORT

Official Website
www.blockaudit.report

E-Mail
team@blockaudit.report

Twitter
https://twitter.com/BlockAudit

Github
https://github.com/blockauditreport