Smart Contract Security Audit

Audit details:

Audited project:	DumplingSwap
Deployer address	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Blockchain:	Binance Smart Chain
Project website:	https://dumplingdefi.finance

May, 2021
TechRate

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

Background

TechRate was commissioned by DumplingSwap to perform an audit of smart contracts:

- *https://bscscan.com/address/0xdD2801cdde911be11B8E2f64fBf5761e7069f83D#contracts*
- *https://bscscan.com/address/0x13F6751ba11337BC67aBBdAd638a56194ee133B8#code*
- *https://bscscan.com/address/0x1422936E4e83aAAE8C3270d9b3F7f8C663637118#contracts*
- *https://bscscan.com/address/0xe2e643B051ABCFBE735b99eE00b2dbFd3a7BD798#contracts*

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts details

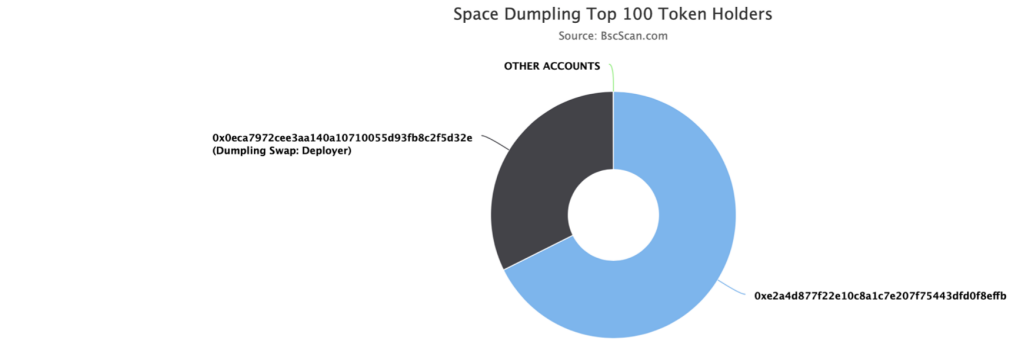
Token contract details for 14.05.2021.

Contract name:	DumplingSwap
Compiler version:	v0.6.12+commit.27d51765
Contract address:	0x13F6751ba11337BC67aBBdAd638a56194ee133B8
Total supply:	340000000000000000000000
Token ticker:	SDUMP
Decimals:	18
Token holders:	2
Transactions count:	3
Top 100 dominance:	100 %
Contract deployer address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Operator:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Contract’s current owner address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e

DumplingSwap token distribution

The top 100 holders collectively own 100.00% (340,000.00 Tokens) of Space Dumpling

Token Total Supply: 340,000.00 Token | Total Token Holders: 2



(A total of 340,000.00 tokens held by the top 100 accounts from the total supply of 340,000.00 token)

MasterChef contract details for 14.05.2021.

Contract name:	MasterChef
Compiler version:	v0.6.12+commit.27d51765
Contract address:	0xe2e643B051ABCFBE735b99eE00b2dbFd3a7BD798
Deployer address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Dev address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Fee address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Dumpling contract address:	0x13f6751ba11337bc67abbdad638a56194ee133b8
Dumpling per block:	10000000000000000000
Dumpling referral:	0x1422936e4e83aaae8c3270d9b3f7f8c663637118
Contract owner address:	0x0eca7972cee3aa140a10710055d93fb8c2f5d32e
Dumpler:	0x334d352ce078c375f168f84ba38695c5141f0cf5
Dumpler mini:	0xbf2ab5811a78ab652185da3028ecb4f4b55f9e12
Pool length:	0
Start block:	7408903
Total alloc point:	0
Bonus multiplier:	1

MasterChef functions outline

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens ($)
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory

  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] mint #
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ ReentrancyGuard
  - [Int] <Constructor> #

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Int] functionStaticCall
  - [Int] functionStaticCall
  - [Int] functionDelegateCall #
  - [Int] functionDelegateCall #
  - [Prv] _verifyCallResult

+ [Lib] SafeMath
  - [Int] tryAdd
  - [Int] trySub
  - [Int] tryMul

  - [Int] tryDiv
  - [Int] tryMod
  - [Int] add
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] mod
  - [Int] sub
  - [Int] div
  - [Int] mod

+ Ownable (Context)
  - [Int] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner

+ [Lib] SafeBEP20
  - [Int] safeTransfer #
  - [Int] safeTransferFrom #
  - [Int] safeApprove #
  - [Int] safeIncreaseAllowance #
  - [Int] safeDecreaseAllowance #
  - [Prv] _callOptionalReturn #

+ [Int] IDumplingReferral
  - [Ext] recordReferral #
  - [Ext] recordReferralCommission #
  - [Ext] getReferrer

+ [Int] IBEP20
  - [Ext] totalSupply
  - [Ext] decimals
  - [Ext] symbol
  - [Ext] name
  - [Ext] getOwner
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ BEP20 (Context, IBEP20, Ownable)
  - [Pub] <Constructor> #
  - [Ext] getOwner

- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Pub] mint #
  - modifiers: onlyOwner
- [Int] _transfer #
- [Int] _mint #
- [Int] _burn #
- [Int] _approve #
- [Int] _burnFrom #

+ SpaceDumplingToken (BEP20)
  - [Pub] <Constructor> #
    - modifiers: BEP20
  - [Pub] mint #
    - modifiers: onlyOwner
  - [Int] _transfer #
    - modifiers: antiWhale
  - [Prv] swapAndLiquify #
    - modifiers: lockTheSwap,transferTaxFree
  - [Prv] swapTokensForEth #
  - [Prv] addLiquidity #
  - [Pub] maxTransferAmount
  - [Pub] isExcludedFromAntiWhale
  - [Ext] <Fallback> ($)
  - [Pub] updateTransferTaxRate #
    - modifiers: onlyOperator
  - [Pub] updateBurnRate #
    - modifiers: onlyOperator
  - [Pub] updateMaxTransferAmountRate #
    - modifiers: onlyOperator
  - [Pub] updateMinAmountToLiquify #
    - modifiers: onlyOperator
  - [Pub] setExcludedFromAntiWhale #
    - modifiers: onlyOperator
  - [Pub] updateSwapAndLiquifyEnabled #
    - modifiers: onlyOperator
  - [Pub] updateLiquidityDivider #
    - modifiers: onlyOperator

  - [Pub] setLiquidityAddress #
    - modifiers: onlyOperator
  - [Pub] updateDumplingSwapRouter #
    - modifiers: onlyOperator
  - [Pub] operator
  - [Pub] transferOperator #
    - modifiers: onlyOperator
  - [Ext] delegates
  - [Ext] delegate #
  - [Ext] delegateBySig #
  - [Ext] getCurrentVotes
  - [Ext] getPriorVotes
  - [Int] _delegate #
  - [Int] _moveDelegates #
  - [Int] _writeCheckpoint #
  - [Int] safe32
  - [Int] getChainId

+ MasterChef (Ownable, ReentrancyGuard)
  - [Pub] <Constructor> #
  - [Ext] poolLength
  - [Pub] add #
    - modifiers: onlyOwner
  - [Pub] set #
    - modifiers: onlyOwner
  - [Pub] getMultiplier
  - [Ext] pendingDumpling
  - [Pub] canHarvest
  - [Pub] massUpdatePools #
  - [Pub] updatePool #
  - [Pub] deposit #
    - modifiers: nonReentrant
  - [Pub] withdraw #
    - modifiers: nonReentrant
  - [Pub] emergencyWithdraw #
    - modifiers: nonReentrant
  - [Int] payOrLockupPendingDumpling #
  - [Int] safeDumplingTransfer #
  - [Pub] setDevAddress #
  - [Pub] setFeeAddress #
  - [Pub] setTeamAddress #
  - [Pub] setDumplerAddress #
  - [Pub] setDumplerMiniAddress #
  - [Pub] updateEmissionRate #
    - modifiers: onlyOwner
  - [Pub] setDumplingReferral #
    - modifiers: onlyOwner

- [Pub] setReferralCommissionRate #
  - modifiers: onlyOwner
- [Int] payReferralCommission #
- [Ext] setStartBlock #
- [Ext] setIsDumplerFeeEnabled #
  - modifiers: onlyOwner

($) = payable function
= non-constant function

Issues Checking Status

№	Issue description.	Checking status
1	Compiler errors.	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.	Low issues
10	Methods execution permissions.	Passed
11	Economy model of the contract.	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.	Medium issues
19	Cross-function race conditions.	Passed
20	Safe Open Zeppelin contracts implementation and usage.	Passed
21	Fallback function security.	Passed

Security Issues

High Severity Issues

No high severity issues found.

Medium Severity Issues

1. Wrong burning

Issue:

There is sending tokens to the dead address in overridden _transfer functions, instead of burning them in token contract.

Recommendation:

There should be a burn instead of sending to the dead address.

Low Severity Issues

1. Block gas limit

Issue:

The updateEmissionRate function can fail due to block gas limit if the pool size is too big.

2. add function issue

Issue:

If some LP token is added to the contract twice using function add, then the total amount of reward dumplingReward in function updatePool will be incorrect.

Recommendation:

Add the mapping from address to bool and check that same address will not be added twice.

Owner privileges

- ❑ Owner can withdraw tokens from the Referral contract, which is useful for withdrawing tokens sent by mistake to the contract.
- ❑ Fee address can change the dev, fee, dumple, dumpler mini and team addresses.
- ❑ Owner can change the referral contract and referral commission rate.
- ❑ Operator can change the burn, tax and max transfer amount rates.
- ❑ Operator can change the dumpling swap router.
- ❑ Owner can change the liquidity address.

Conclusion

Smart contracts contain medium severity issues. Liquidity will be added to the operator or liquidity address, but it will be better to add the liquidity to the contract to restrict the management of the lp tokens within the scope of the contract's business logic.

```
function addLiquidity(uint256 tokenAmount⬆, uint256 ethAmount⬆) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(dumplingSwapRouter), tokenAmount⬆);
    address liquidityAddressTarget = operator();
    if (liquidityAddress != BURN_ADDRESS) {
        liquidityAddressTarget = liquidityAddress;
    }
    // add the liquidity
    dumplingSwapRouter.addLiquidityETH{value: ethAmount⬆}(
        address(this),
        tokenAmount⬆,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityAddressTarget,
        block.timestamp
    );
}
```

Techrate 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.