Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

December, 2021

Audit Details

Audited project

The Essential Coin

Deployer address

0xd1e78c9d59746c4f9673038b45faa999e586ab75

Client contacts:

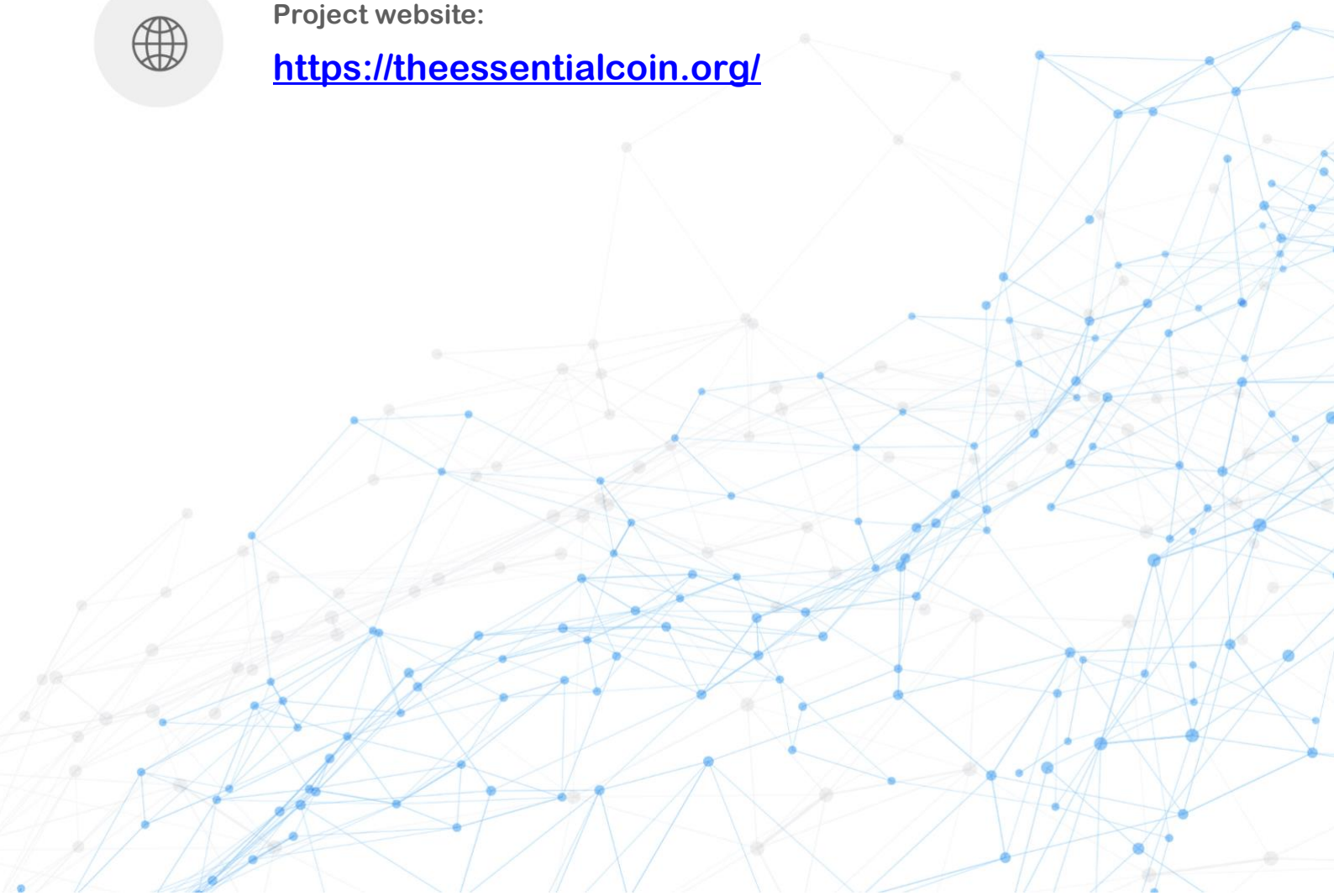
The Essential Coin team

Blockchain

Binance Smart Chain

Project website:

https://theessentialcoin.org/



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 The Essential Coin to perform an audit of smart contracts:
https://bscscan.com/address/0x4c48cca6153Db911002F965D22fdeFcD95f33BE9#code

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

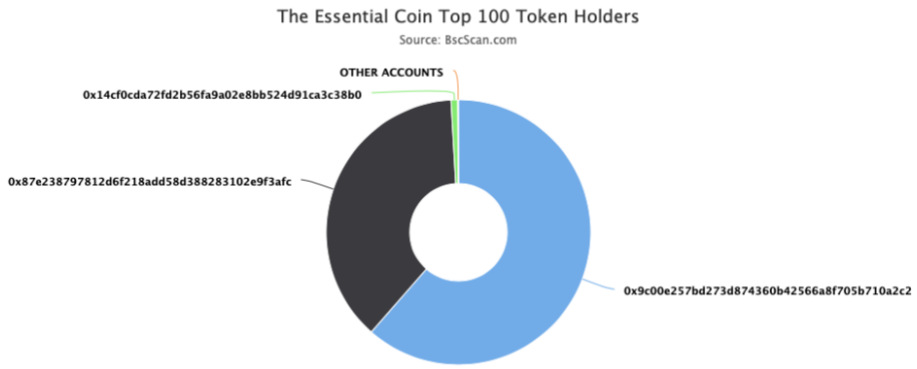
Token contract details for 14.12.2021

Contract name	The Essential Coin
Contract address	0x4c48cca6153Db911002F965D22fdeFcD95f33BE9
Total supply	1,000,000,000,000,000
Token ticker	ESC
Decimals	18
Token holders	3
Transactions count	8
Top 100 holders dominance	99.92%
Liquidity fee	4
Tax fee	5
Total fees	410000000000000000000000000000
PCS V2 pair	0xf8abe75aa59df74d15bb50ae36219fd03d70a550
Contract deployer address	0xd1e78c9d59746c4f9673038b45faa999e586ab75
Contract's current owner address	0x87e238797812d6f218add58d388283102e9f3afc

The Essential Coin Token Distribution

The top 100 holders collectively own 99.92% (999,179,056,651,421.00 Tokens) of The Essential Coin

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 3

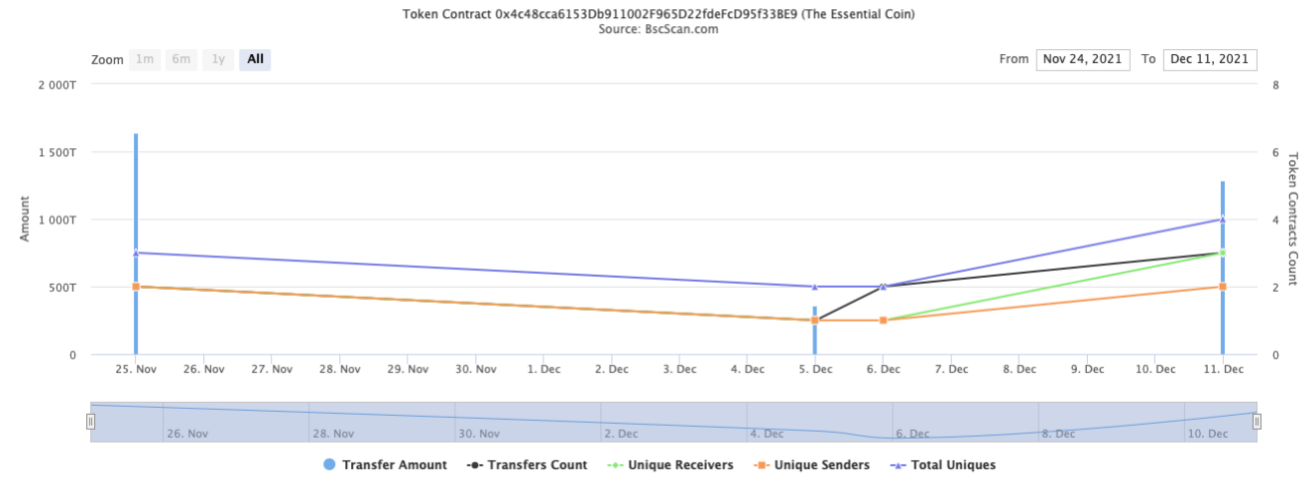


(A total of 999,179,056,651,421.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

The Essential Coin Contract Interaction Details

Time Series: Token Contract Overview

Thu 25, Nov 2021 - Sat 11, Dec 2021



The Essential Coin Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	0x9c00e257bd273d874360b42566a8f705b710a2c2	614,560,000,000,000	61.4560%
2	0x87e238797812d6f218add58d388283102e9f3afc	376,411,038,188,501.360868711243300878	37.6411%
3	0x14cf0cda72fd2b56fa9a02e8bb524d91ca3c38b0	8,208,018,462,919.835012205729678161	0.8208%

Contract functions details

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod

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

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] _functionCallWithValue #

+ [Lib] SafeERC20
  - [Int] safeTransfer #
  - [Int] safeTransferFrom #
  - [Int] safeApprove #
  - [Int] safeIncreaseAllowance #
  - [Int] safeDecreaseAllowance #
  - [Prv] _callOptionalReturn #

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

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

+ [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 #

+ Token (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] isExcludedFromReward
  - [Pub] totalFees
  - [Pub] deliver #
  - [Pub] reflectionFromToken

- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Pub] excludeFromFee #
  - modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setAllFeePercent #
  - modifiers: onlyOwner
- [Pub] buyBackUpperLimitAmount
- [Ext] setBuybackUpperLimit #
  - modifiers: onlyOwner
- [Ext] setMaxTxPercent #
  - modifiers: onlyOwner
- [Ext] setMaxWalletPercent #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Ext] setFeeWallet #
  - modifiers: onlyOwner
- [Ext] <Fallback> ($)
- [Prv] _reflectFee #
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Prv] _takeLiquidity #
- [Prv] calculateTaxFee
- [Prv] calculateLiquidityFee
- [Prv] removeAllFee #
- [Prv] restoreAllFee #
- [Pub] isExcludedFromFee
- [Prv] _approve #
- [Prv] _transfer #
- [Prv] swapAndLiquify #
  - modifiers: lockTheSwap
- [Prv] buyBackTokens #
  - modifiers: lockTheSwap
- [Prv] swapTokensForBNB #
- [Prv] swapBNBForTokens #
- [Prv] addLiquidity #
- [Prv] _tokenTransfer #
- [Prv] _transferStandard #
- [Prv] _transferToExcluded #
- [Prv] _transferFromExcluded #
- [Prv] _transferBothExcluded #
- [Prv] _tokenTransferNoFee #
- [Pub] recoverBEP20 #
  - 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.	Passed
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

No medium severity issues found.

Low Severity Issues

1. Out of gas

Issue:

- The function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function includeInReward(address account⬆) external onlyOwner() {
    require(_isExcluded[account⬆], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account⬆) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account⬆] = 0;
            _isExcluded[account⬆] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The function `_getCurrentSupply` also uses the loop for evaluating total supply. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation:

Check that the excluded array length is not too big.

Notes:

- There is sending contract balance to dead address instead of real burn, using decreasing total supply.

Owner privileges (In the period when the owner is not renounced)

- Owner can change fees.

```
ftrace | funcSig
function setAllFeePercent(uint8 taxFee↑, uint8 liquidityFee↑, uint8 burnFee↑, uint8 walletFee↑, uint8 buybackFee↑) external onlyOwner() {
    require(taxFee↑ >= 0 && taxFee↑ <=maxTaxFee,"TF err");
    require(liquidityFee↑ >= 0 && liquidityFee↑ <=maxLiqFee,"LF err");
    require(burnFee↑ >= 0 && burnFee↑ <=maxBurnFee,"BF err");
    require(walletFee↑ >= 0 && walletFee↑ <=maxWalletFee,"WF err");
    require(buybackFee↑ >= 0 && buybackFee↑ <=maxBuybackFee,"BBF err");
    _taxFee = taxFee↑;
    _liquidityFee = liquidityFee↑;
    _burnFee = burnFee↑;
    _buybackFee = buybackFee↑;
    _walletFee = walletFee↑;
}
```

- Owner can change the maximum transaction amount and maximum wallet amount.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxPercent↑) external onlyOwner() {
    require(maxTxPercent↑ >= minMxTxPercentage && maxTxPercent↑ <=100,"err");
    _maxTxAmount = _tTotal.mul(maxTxPercent↑).div(
        10**2
    );
}

ftrace | funcSig
function setMaxWalletPercent(uint256 maxWalletPercent↑) external onlyOwner() {
    require(maxWalletPercent↑ >= minMxWalletPercentage && maxWalletPercent↑ <=100,"err");
    _maxWalletAmount = _tTotal.mul(maxWalletPercent↑).div(
        10**2
    );
}
```

- Owner can change buyBackUpperLimit.

```
ftrace | funcSig
function setBuybackUpperLimit(uint256 buyBackLimit↑) external onlyOwner() {
    buyBackUpperLimit = buyBackLimit↑ * 10**18;
}
```

- Owner can exclude from the fee.

```
    function excludeFromFee(address account↑) public onlyOwner {
        _isExcludedFromFee[account↑] = true;
    }
```

- Owner can change fee wallet.

```
ftrace | funcSig
function setFeeWallet(address payable newFeeWallet⬆) external onlyOwner {
    require(newFeeWallet⬆ != address(0), "ZERO ADDRESS");
    feeWallet = newFeeWallet⬆;
}
```

- Owner can withdraw ERC tokens except native.

```
ftrace | funcSig
function recoverBEP20(address tokenAddress⬆, uint256 tokenAmount⬆) public onlyOwner {
    // do not allow recovering self token
    require(tokenAddress⬆ != address(this), "Self withdraw");
    IERC20(tokenAddress⬆).transfer(owner(), tokenAmount⬆);
}
```

- Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = now + time;
    emit OwnershipTransferred(_owner, address(0));
}

//Unlocks the contract for owner when _lockTime is exceeds
function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(now > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

Conclusion

Smart contracts contain low severity issues! Liquidity pair contract's security is not checked due to out of scope.

Liquidity locking details provided by the team:
https://www.pinksale.finance/#/pinklock/detail/0xF8abE75aA59DF74d15bb50Ae36219Fd03D70A550?chain=BSC

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.