Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

June, 2021

Audit Details

Audited project

WenLambo

Deployer address

0x357B174d3690998845c0A5D3B2762E8c600BB814

Client contacts:

WenLambo team

Blockchain

Binance Smart Chain

Project website:

https://wenlambo.finance/

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 WenLambo to perform an audit of smart contracts:
https://bscscan.com/address/0xd8a31016cd7da048ca21ffe04256c6d08c3a2251#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

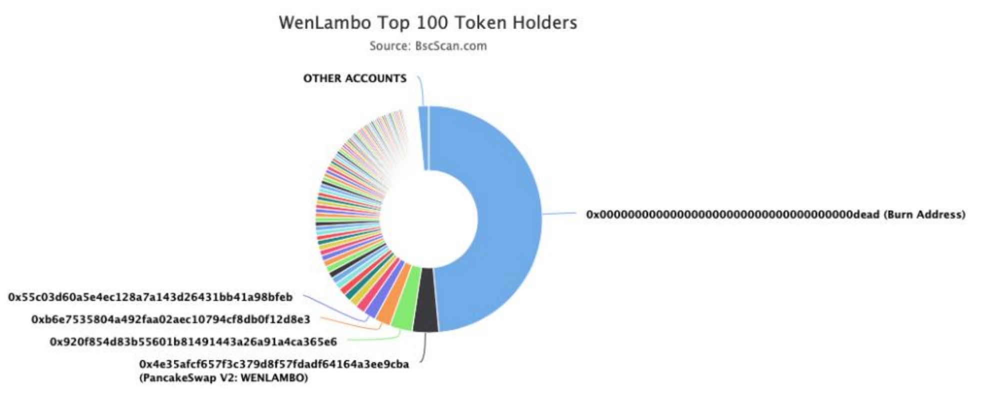
Token contract details for 24.06.2021

Contract name	WenLambo
Contract address	0xd8A31016cD7da048ca21FFE04256C6d08C3A2251
Total supply	99,257,840,116,030.345435
Token ticker	WENLAMBO
Decimals	18
Token holders	10,788
Transactions count	104,830
Top 100 holders dominance	98.44%
Tax fee	400
Total fees	11775815149918633784948967073567
Contract deployer address	0x357B174d3690998845c0A5D3B2762E8c600BB814
Contract's current owner address	0xb6e7535804a492faa02aec10794cf8db0f12d8e3

WenLambo Token Distribution

The top 100 holders collectively own 98.44% (97,710,593,216,352.70 Tokens) of WenLambo

Token Total Supply: 99,257,840,116,030.35 Token | Total Token Holders: 10,788

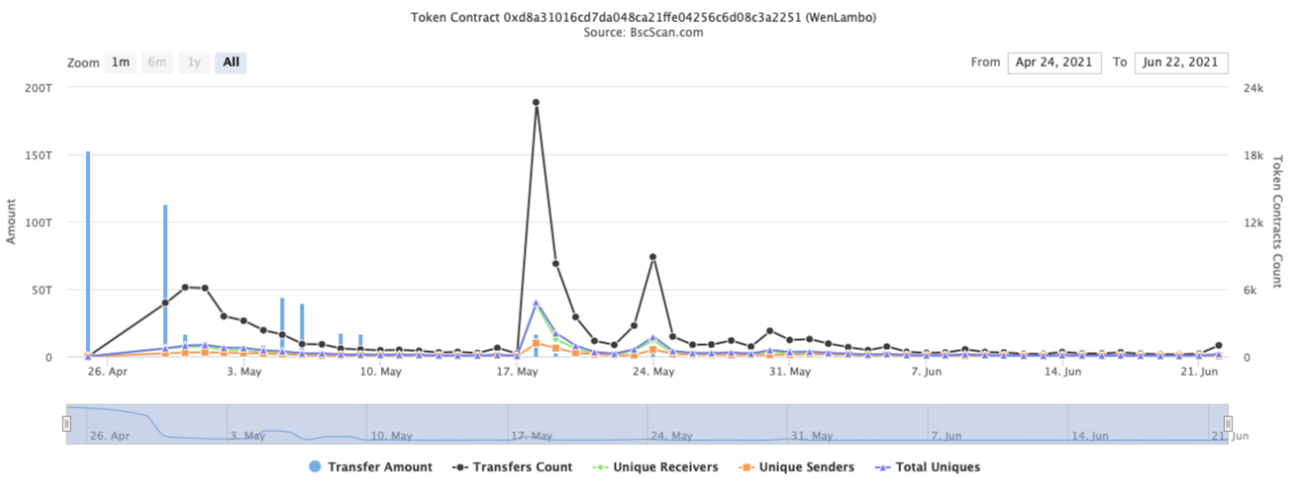


(A total of 97,710,593,216,352.70 tokens held by the top 100 accounts from the total supply of 99,257,840,116,030.35 token)

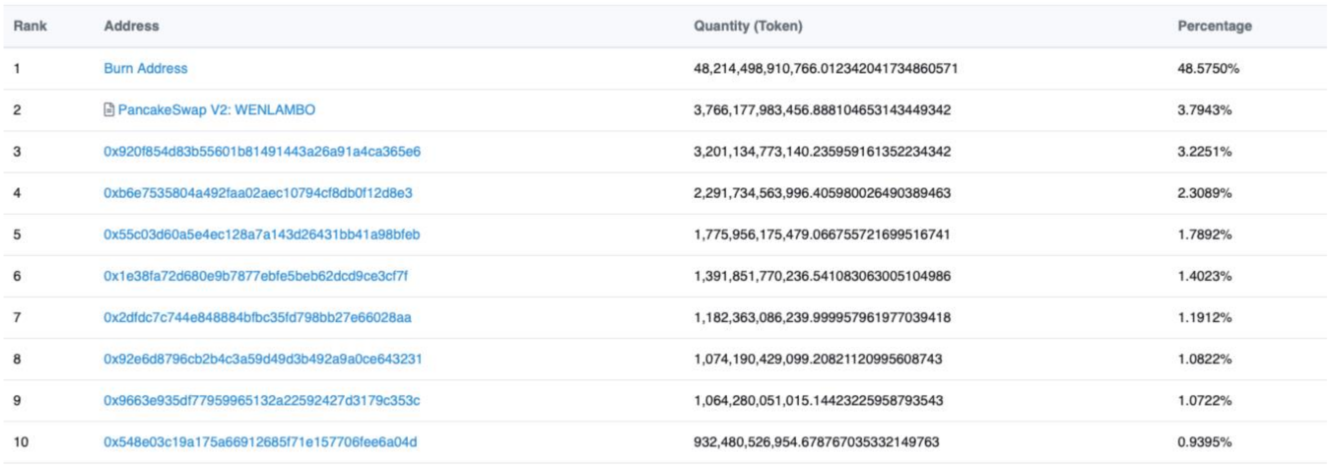
WenLambo Contract Interaction Details

Time Series: Token Contract Overview

Sun 25, Apr 2021 - Tue 22, Jun 2021



WenLambo Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	Burn Address	48,214,498,910,766.012342041734860571	48.5750%
2	PancakeSwap V2: WENLAMBO	3,766,177,983,456.888104653143449342	3.7943%
3	0x920f854d83b55601b81491443a26a91a4ca365e6	3,201,134,773,140.235959161352234342	3.2251%
4	0xb6e7535804a492faa02aec10794cf8db0f12d8e3	2,291,734,563,996.405980026490389463	2.3089%
5	0x55c03d60a5e4ec128a7a143d26431bb41a98bfeb	1,775,956,175,479.066755721699516741	1.7892%
6	0x1e38fa72d680e9b7877ebfe5beb62dcd9ce3cf7f	1,391,851,770,236.541083063005104986	1.4023%
7	0x2dfdc7c744e848884bfbc35fd798bb27e66028aa	1,182,363,086,239.999957961977039418	1.1912%
8	0x92e6d8796cb2b4c3a59d49d3b492a9a0ce643231	1,074,190,429,099.208211209956087431	1.0822%
9	0x9663e935df77959965132a22592427d3179c353c	1,064,280,051,015.144232259587935433	1.0722%
10	0x548e03c19a175a66912685f71e157706fee6a04d	932,480,526,954.678767035332149763	0.9395%

Contract functions details

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

+ [Int] IBEP20
  - [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

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

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

+  CoinToken (Context, IBEP20, 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] isExcluded
  - [Pub] isCharity
  - [Pub] totalFees
  - [Pub] totalBurn
  - [Pub] totalCharity
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Ext] excludeAccount #
    - modifiers: onlyOwner
  - [Ext] includeAccount #
    - modifiers: onlyOwner
  - [Ext] setAsCharityAccount #
    - modifiers: onlyOwner
  - [Pub] burn #
  - [Pub] updateFee #
    - modifiers: onlyOwner
  - [Int] _burn #
  - [Pub] mint #
    - modifiers: onlyOwner
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] _transferStandard #
  - [Prv] _standardTransferContent #
  - [Prv] _transferToExcluded #
  - [Prv] _excludedFromTransferContent #
  - [Prv] _transferFromExcluded #
  - [Prv] _excludedToTransferContent #
  - [Prv] _transferBothExcluded #
  - [Prv] _bothTransferContent #
  - [Prv] _reflectFee #
  - [Prv] _getValues
  - [Prv] _getTBasics
  - [Prv] getTTransferAmount
  - [Prv] _getRBasics
  - [Prv] _getRTransferAmount
  - [Prv] _getRate
  - [Prv] _getCurrentSupply
  - [Prv] _sendToCharity #
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Prv] _getTaxFee

($) = 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.	High issues
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

1. Wrong burn and mint

Issue:

- In burn and mint functions there are wrong values adding because of not converting _value. _rOwner and _tTotal show balances in different modes and same values will be added / subtracted to them, which will make it wrong.

Recommendation:

Please check if the addresses are included in reward or not and add the values correctly by multiplying by the rate.

Medium Severity Issues

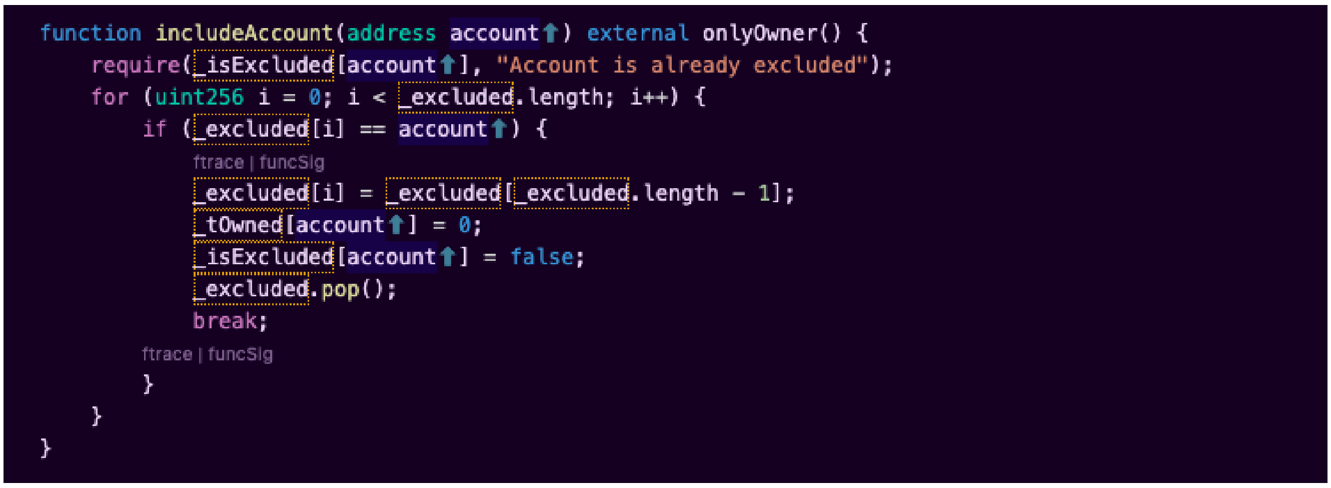
No medium severity issues found.

Low Severity Issues

2. Out of gas

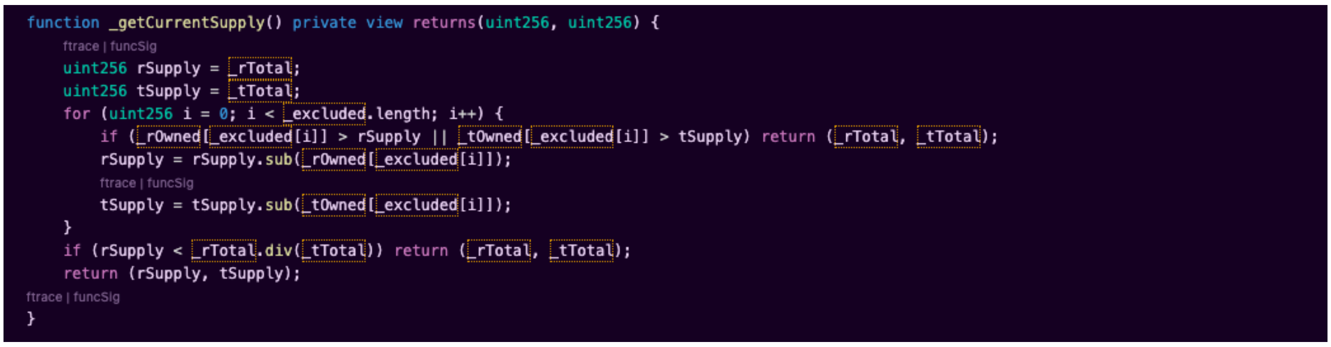
Issue:

- The function includeAccount() 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 includeAccount(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.



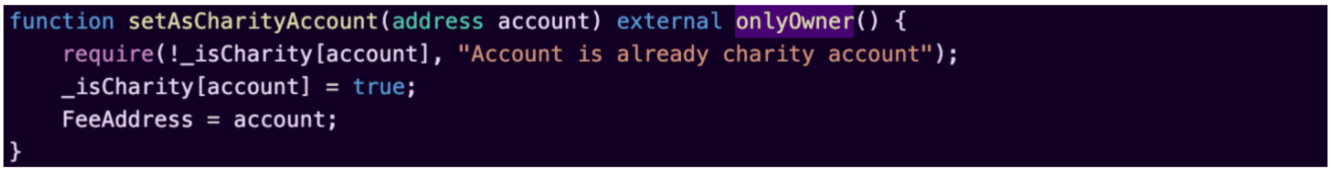
```
function _getCurrentSupply() private view returns(uint256, uint256) {
    ftrace | funcSig
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal,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

Recommendation:

Check that the excluded array length is not too big.

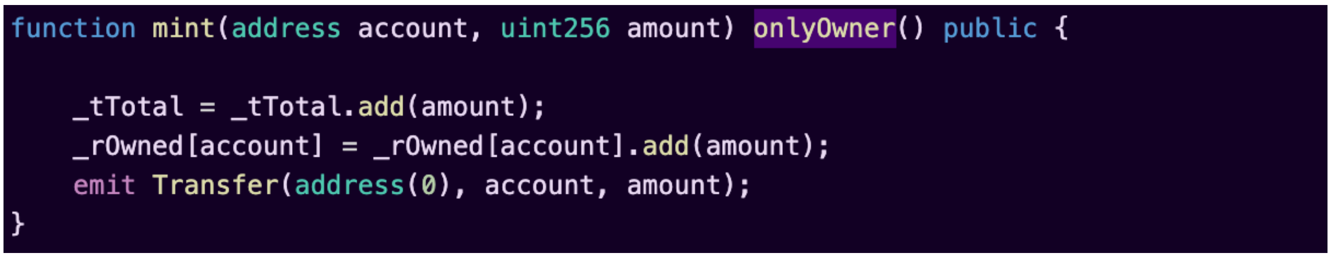
Owner privileges (In the period when the owner is not renounced)

- Owner can change charity address.



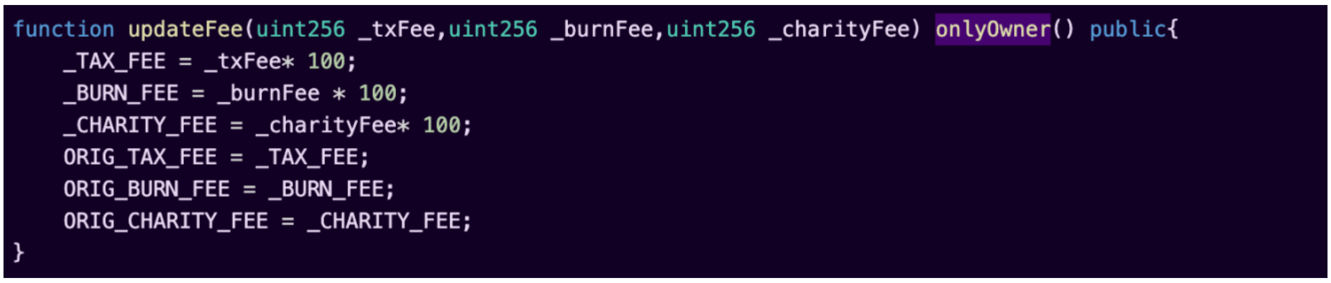
```
function setAsCharityAccount(address account) external onlyOwner() {
    require(!_isCharity[account], "Account is already charity account");
    _isCharity[account] = true;
    FeeAddress = account;
}
```

- Owner can mint.



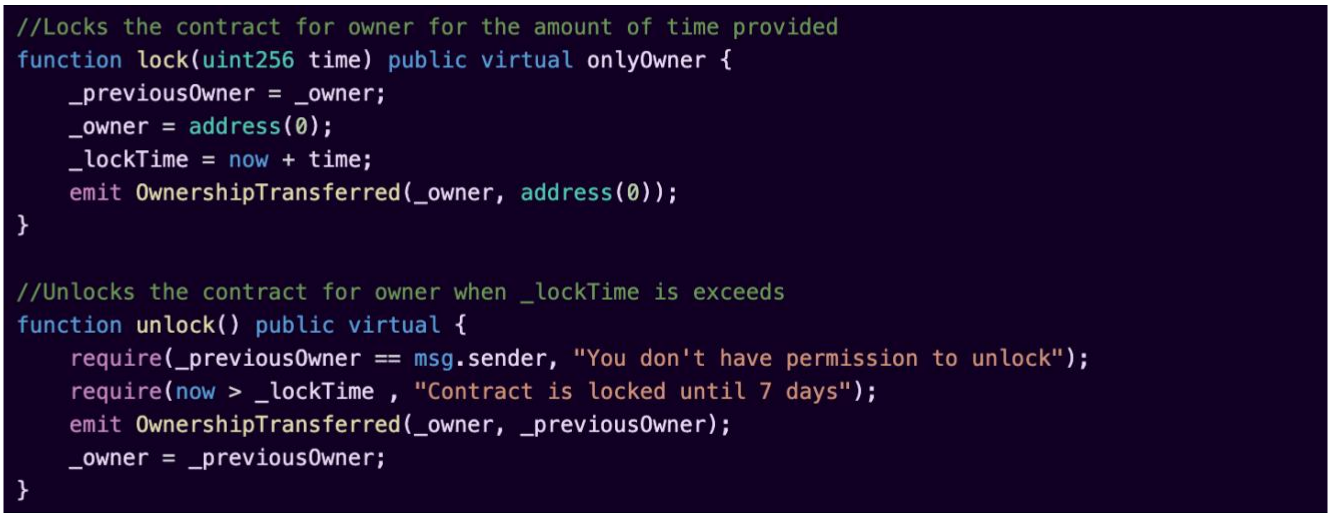
```
function mint(address account, uint256 amount) onlyOwner() public {

    _tTotal = _tTotal.add(amount);
    _rOwned[account] = _rOwned[account].add(amount);
    emit Transfer(address(0), account, amount);
}
```

- Owner can change fees.



```
function updateFee(uint256 _txFee,uint256 _burnFee,uint256 _charityFee) onlyOwner() public{
    _TAX_FEE = _txFee* 100;
    _BURN_FEE = _burnFee * 100;
    _CHARITY_FEE = _charityFee* 100;
    ORIG_TAX_FEE = _TAX_FEE;
    ORIG_BURN_FEE = _BURN_FEE;
    ORIG_CHARITY_FEE = _CHARITY_FEE;
}
```

- 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 high severity issues! Liquidity pair contract's security is not checked due to out of scope.

Liquidity locking details provided by the team:
http://dxsale.app/app/pages/dxlockview?id=0&add=0x0A8543f74bb324DD32E8BCB7063E317aD0A015Bc&type=lplock&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.