Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

December, 2021

Audit Details

Audited project

MeglaDoge

Deployer address

0x9e43790c5f7e72e1d6cf5a08e01f52480eed6817

Client contacts:

MeglaDoge team

Blockchain

Binance Smart Chain

Project website:

megladoge.com

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 MeglaDoge to perform an audit of smart contracts:

https://bscscan.com/address/0x87c55991dd7c0f946dfc4ffcc513b61758096966#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.

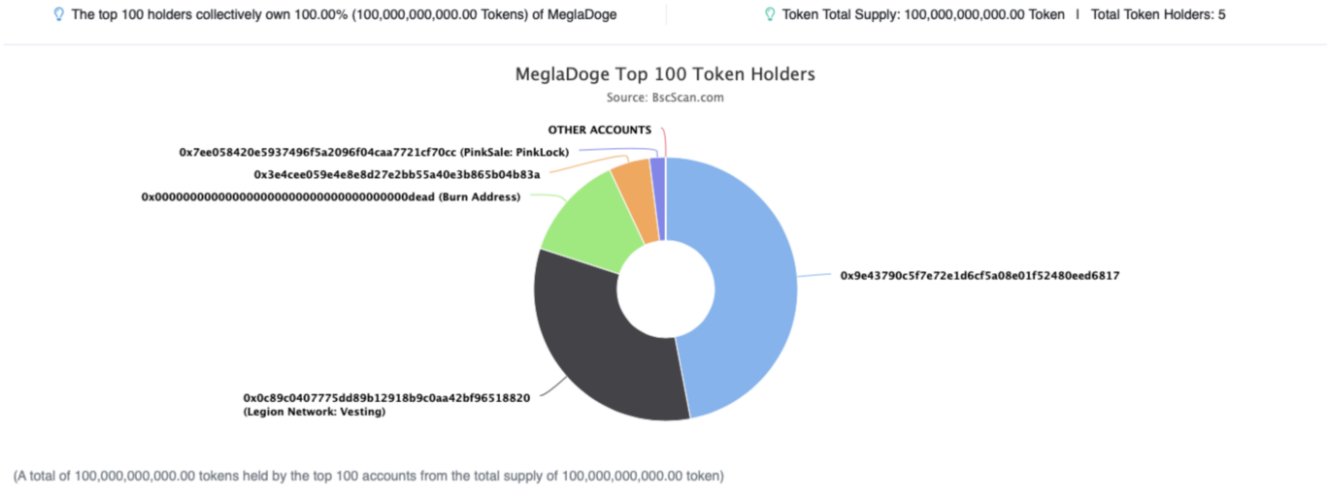
The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 04.12.2021

Contract name	MeglaDoge
Contract address	0x87C55991Dd7C0F946Dfc4ffcc513b61758096966
Total supply	100,000,000,000
Token ticker	MGD
Decimals	18
Token holders	5
Transactions count	9
Top 100 holders dominance	100.00%
Contract deployer address	0x9e43790c5f7e72e1d6cf5a08e01f52480eed6817
Contract's current owner address	0x9e43790c5f7e72e1d6cf5a08e01f52480eed6817

MeglaDoge Token Distribution



The top 100 holders collectively own 100.00% (100,000,000,000.00 Tokens) of MeglaDoge

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 5

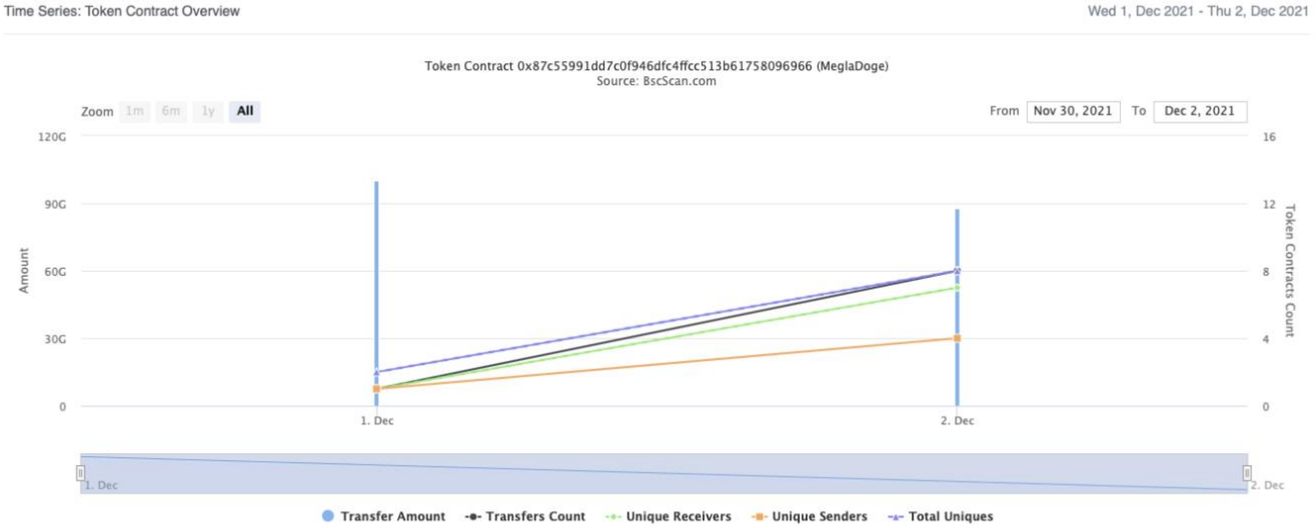
MeglaDoge Top 100 Token Holders

Source: BscScan.com

OTHER ACCOUNTS
0x7ee058420e5937496f5a2096f04caa7721cf70cc (PinkSale: PinkLock)
0x3e4cee059e4e8e8d27e2bb55a40e3b865b04b83a
0x000000000000000000000000000000000000dead (Burn Address)
0x9e43790c5f7e72e1d6cf5a08e01f52480eed6817
0x0c89c0407775dd89b12918b9c0aa42bf96518820 (Legion Network: Vesting)

(A total of 100,000,000,000.00 tokens held by the top 100 accounts from the total supply of 100,000,000,000.00 token)

MeglaDoge Contract Interaction Details



Time Series: Token Contract Overview

Wed 1, Dec 2021 - Thu 2, Dec 2021

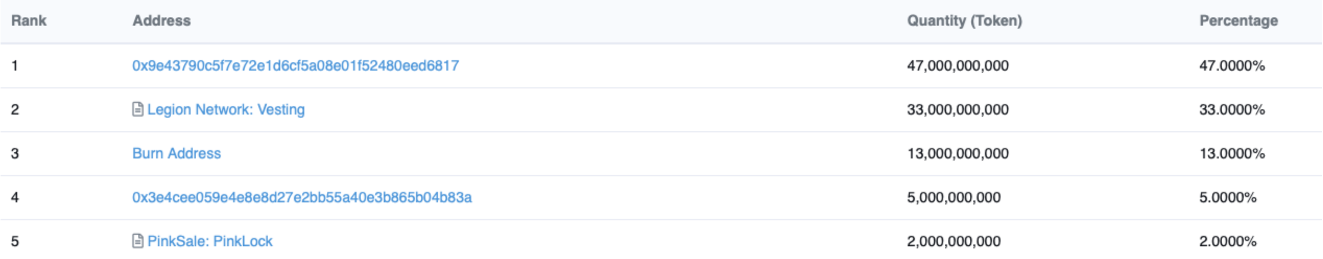
Token Contract 0x87c55991dd7c0f946dfc4ffcc513b61758096966 (MeglaDoge)

Source: BscScan.com

Zoom 1m 6m 1y All

From Nov 30, 2021 To Dec 2, 2021

MeglaDoge Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	0x9e43790c5f7e72e1d6cf5a08e01f52480eed6817	47,000,000,000	47.0000%
2	📄 Legion Network: Vesting	33,000,000,000	33.0000%
3	Burn Address	13,000,000,000	13.0000%
4	0x3e4cee059e4e8e8d27e2bb55a40e3b865b04b83a	5,000,000,000	5.0000%
5	📄 PinkSale: PinkLock	2,000,000,000	2.0000%

Contract functions details

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

 + [Int] IPancakeFactory
  - [Ext] createPair #

 + [Int] IPancakeRouter
  - [Ext] addLiquidityETH ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #
  - [Ext] factory
  - [Ext] WETH

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

 +  MeglaDoge (IBEP20, Ownable)
  - [Pub] ChangeMarketingWallet #
  - [Prv] _isTeam
  - [Pub] <Constructor> #
  - [Prv] _transfer #
  - [Prv] _taxedTransfer #
  - [Prv] _getStartTax
  - [Prv] _calculateFee
  - [Prv] _feelessTransfer #
  - [Pub] setSwapTreshold #
    - modifiers: onlyTeam
  - [Pub] SetOverLiquifiedTreshold #
    - modifiers: onlyTeam
  - [Pub] SetTaxes #
    - modifiers: onlyTeam
  - [Pub] isOverLiquified
  - [Prv] _swapContractToken #
    - modifiers: lockTheSwap
  - [Prv] _swapTokenForBNB #
  - [Prv] _addLiquidity #
  - [Pub] getLiquidityReleaseTimeInSeconds
  - [Pub] getBurnedTokens
```

- [Pub] SetAMM #
  - modifiers: onlyTeam
- [Pub] SwitchManualSwap #
  - modifiers: onlyTeam
- [Pub] SwapContractToken #
  - modifiers: onlyTeam
- [Pub] ExcludeAccountFromFees #
  - modifiers: onlyTeam
- [Pub] SetupEnableTrading #
  - modifiers: onlyTeam
- [Pub] limitLiquidityReleaseTo20Percent #
  - modifiers: onlyTeam
- [Pub] LockLiquidityForSeconds #
  - modifiers: onlyTeam
- [Prv] _prolongLiquidityLock #
- [Pub] LiquidityRelease #
  - modifiers: onlyTeam
- [Ext] <Fallback> ($)
- [Ext] getOwner
- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Pub] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Prv] _approve #
- [Ext] transferFrom #
- [Ext] increaseAllowance #
- [Ext] decreaseAllowance #

($) = 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.	Passed
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

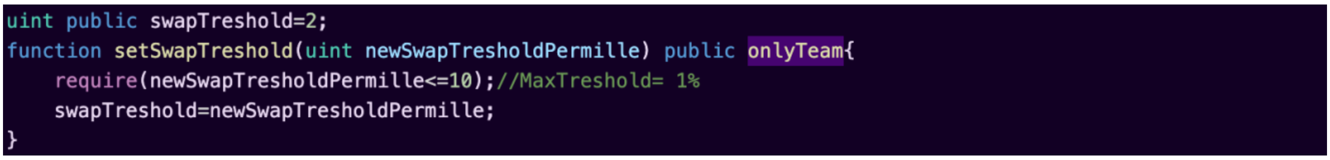
No medium severity issues found.

Low Severity Issues

No low severity issues found.

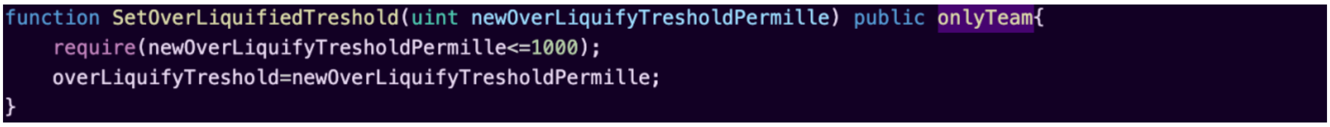
Owner privileges (In the period when the owner is not renounced)

- Owner and wallet address can change swapThreshold value.



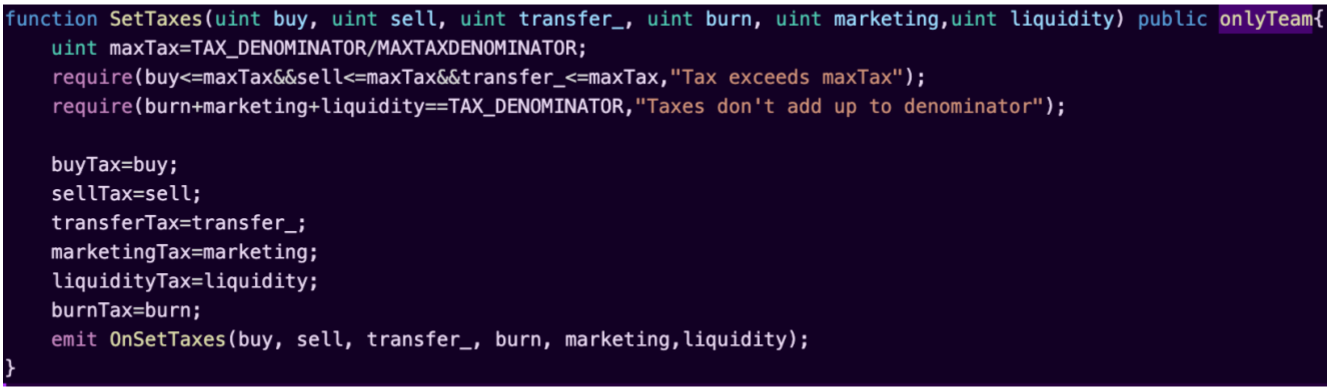
```
uint public swapTreshold=2;
function setSwapTreshold(uint newSwapTresholdPermille) public onlyTeam{
    require(newSwapTresholdPermille<=10);//MaxTreshold= 1%
    swapTreshold=newSwapTresholdPermille;
}
```

- Owner and wallet address can change overLiquifyThreshold value.



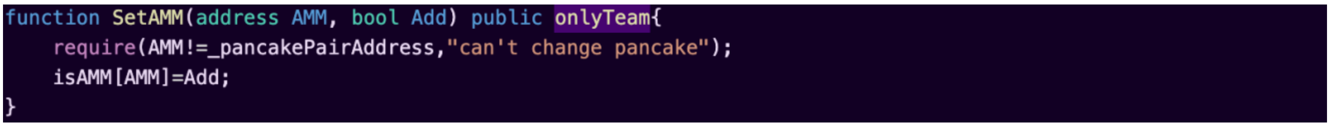
```
function SetOverLiquifiedTreshold(uint newOverLiquifyTresholdPermille) public onlyTeam{
    require(newOverLiquifyTresholdPermille<=1000);
    overLiquifyTreshold=newOverLiquifyTresholdPermille;
}
```

- Owner and wallet address can change buy, sell, transfer, burn, marketing and liquidity taxes.



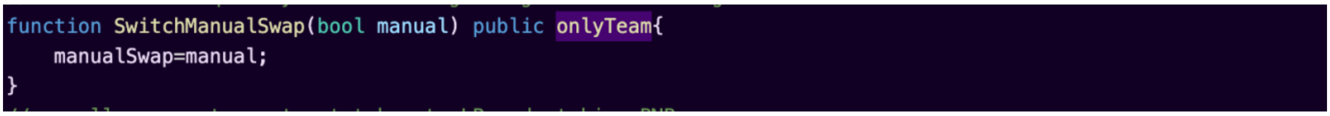
```
function SetTaxes(uint buy, uint sell, uint transfer_, uint burn, uint marketing,uint liquidity) public onlyTeam{
    uint maxTax=TAX_DENOMINATOR/MAXTAXDENOMINATOR;
    require(buy<=maxTax&&sell<=maxTax&&transfer_<=maxTax,"Tax exceeds maxTax");
    require(burn+marketing+liquidity==TAX_DENOMINATOR,"Taxes don't add up to denominator");

    buyTax=buy;
    sellTax=sell;
    transferTax=transfer_;
    marketingTax=marketing;
    liquidityTax=liquidity;
    burnTax=burn;
    emit OnSetTaxes(buy, sell, transfer_, burn, marketing,liquidity);
}
```

- Owner and wallet address can include in and exclude from buy / sell taxes.



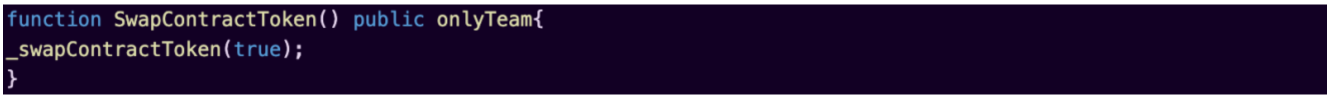
```
function SetAMM(address AMM, bool Add) public onlyTeam{
    require(AMM!=_pancakePairAddress,"can't change pancake");
    isAMM[AMM]=Add;
}
```

- Owner and wallet address can enable / disable automatic swap.



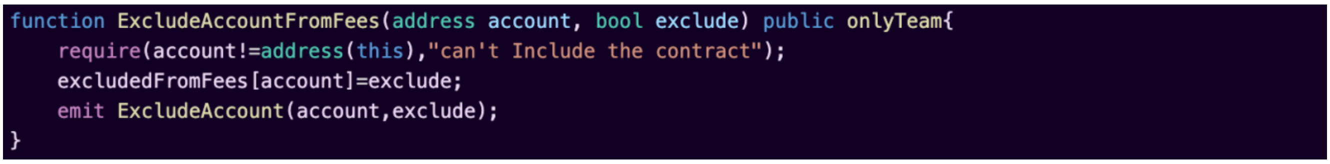
```
function SwitchManualSwap(bool manual) public onlyTeam{
    manualSwap=manual;
}
```

- Owner and wallet address can swap contract tokens without limit.



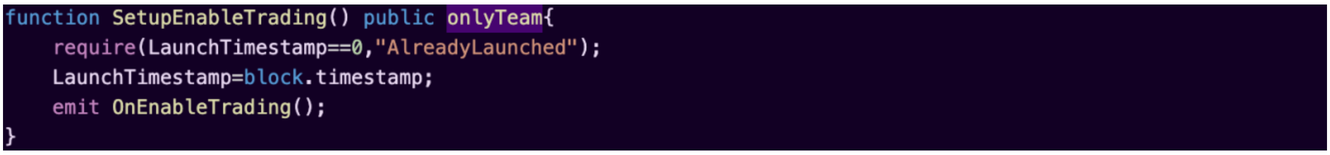
```
function SwapContractToken() public onlyTeam{
_swapContractToken(true);
}
```

- Owner and wallet address can include in and exclude from fees.



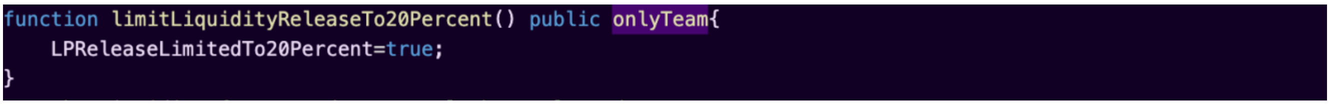
```
function ExcludeAccountFromFees(address account, bool exclude) public onlyTeam{
    require(account!=address(this),"can't Include the contract");
    excludedFromFees[account]=exclude;
    emit ExcludeAccount(account,exclude);
}
```

- Owner and wallet address can enable trading.



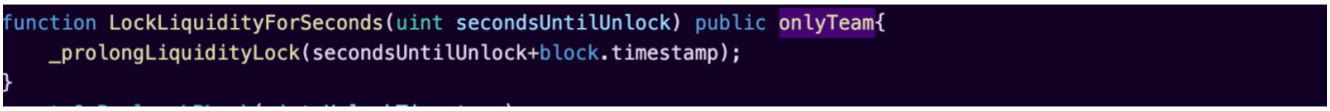
```
function SetupEnableTrading() public onlyTeam{
    require(LaunchTimestamp==0,"AlreadyLaunched");
    LaunchTimestamp=block.timestamp;
    emit OnEnableTrading();
}
```

- Owner and wallet address can change liquidity release to 20%.



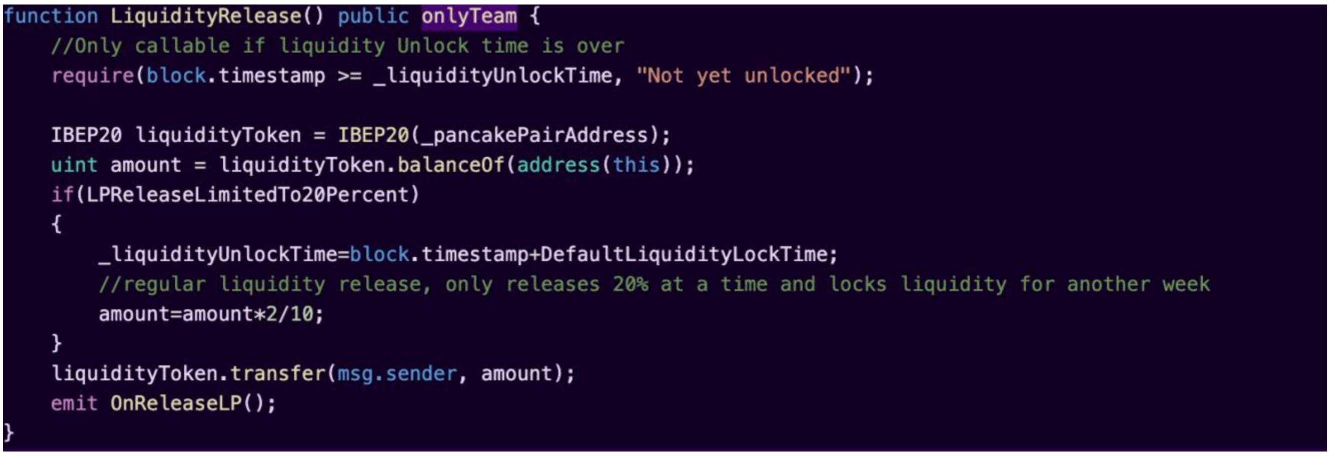
```
function limitLiquidityReleaseTo20Percent() public onlyTeam{
    LPReleaseLimitedTo20Percent=true;
}
```

- Owner and wallet address can lock liquidity for a time.



```
function LockLiquidityForSeconds(uint secondsUntilUnlock) public onlyTeam{
    _prolongLiquidityLock(secondsUntilUnlock+block.timestamp);
}
```

- Owner and wallet address can release liquidity tokens once unlock time is over.



```
function LiquidityRelease() public onlyTeam {
    //Only callable if liquidity Unlock time is over
    require(block.timestamp >= _liquidityUnlockTime, "Not yet unlocked");

    IBEP20 liquidityToken = IBEP20(_pancakePairAddress);
    uint amount = liquidityToken.balanceOf(address(this));
    if(LPReleaseLimitedTo20Percent)
    {
        _liquidityUnlockTime=block.timestamp+DefaultLiquidityLockTime;
        //regular liquidity release, only releases 20% at a time and locks liquidity for another week
        amount=amount*2/10;
    }
    liquidityToken.transfer(msg.sender, amount);
    emit OnReleaseLP();
}
```

Conclusion

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

Liquidity locking details NOT provided by the team.

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.