Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

June, 2021

Audit Details

Audited project

EARNDOGE

Deployer address

0x91711a669D9Ff755e863dbADF98d42e609412289

Client contacts:

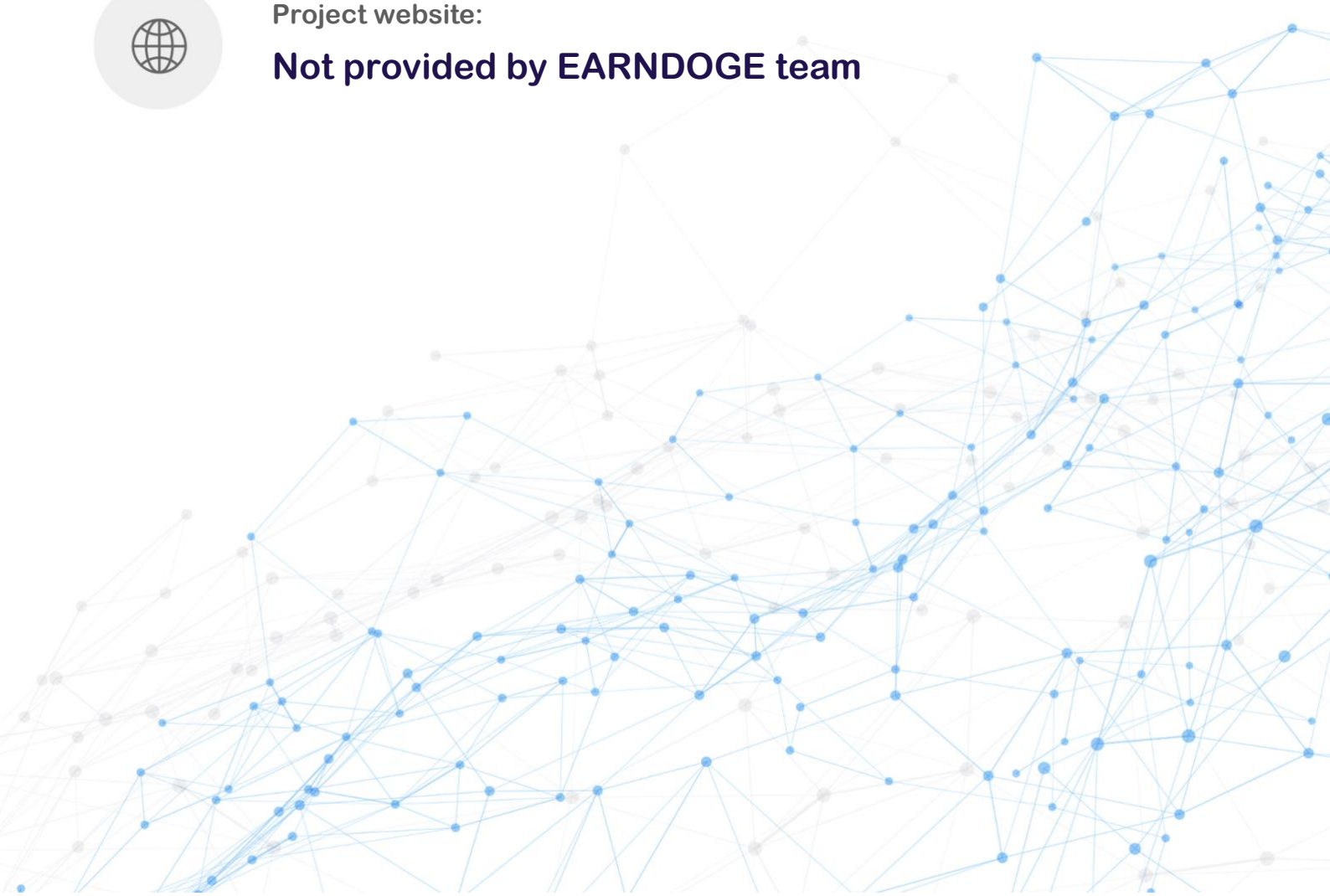
EARNDOGE team

Blockchain

Binance Smart Chain

Project website:

Not provided by EARNDOGE team



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 EARNDOGE to perform an audit of smart contracts:

https://bscscan.com/address/0x3ee4c28ec61e3446289de4c9124866fccf9b9511#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

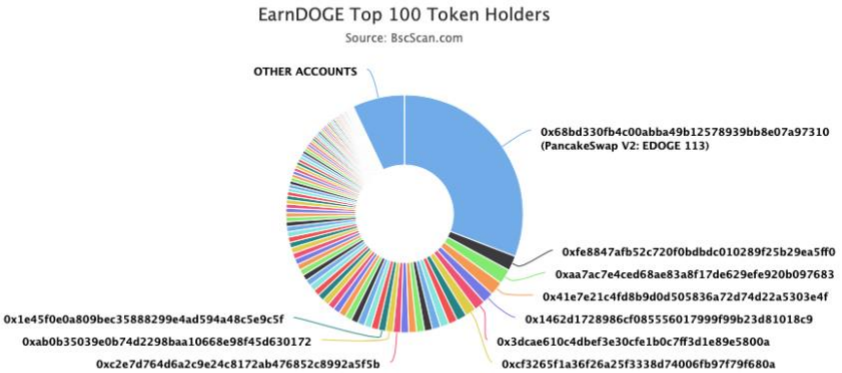
Token contract details for 17.06.2021

Contract name	EARNDOGE
Contract address	0x3eE4c28EC61e3446289De4c9124866FCCf9b9511
Total supply	98,901,608.137005
Token ticker	EDOGE
Decimals	9
Token holders	457
Transactions count	4,850
Top 100 holders dominance	92.90%
Liquidity fee	60
Tax fee	12
Total payots	33521827403209996284
Team wallet	0x4ca0531621ca33b64a3c147e4ee0fca6cb827981
Contract deployer address	0x91711a669D9Ff755e863dbADF98d42e609412289
Contract's current owner address	0x91711a669d9ff755e863dbadf98d42e609412289

EARNDOGE Token Distribution

The top 100 holders collectively own 92.90% (91,876,493.81 Tokens) of EarnDOGE

Token Total Supply: 98,901,608.14 Token | Total Token Holders: 457

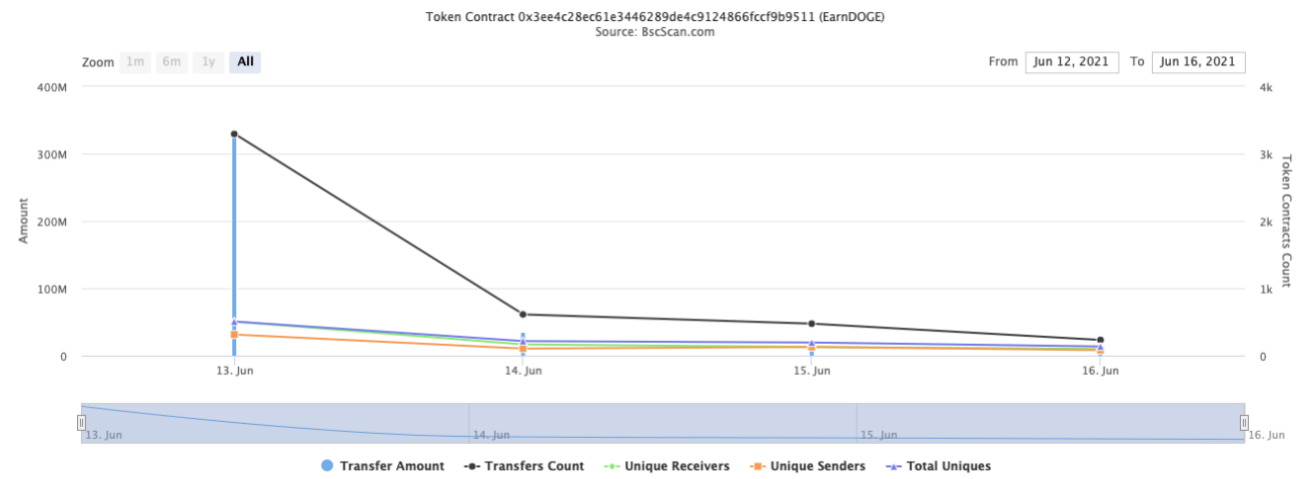


(A total of 91,876,493.81 tokens held by the top 100 accounts from the total supply of 98,901,608.14 token)

EARNDOGE Contract Interaction Details

Time Series: Token Contract Overview

Sun 13, Jun 2021 - Wed 16, Jun 2021



EARNDOGE Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: EDOGE 113	30,509,391.281308876	30.8482%
2	0xfe8847afb52c720f0bdbdc010289f25b29ea5ff0	1,999,945.492419506	2.0222%
3	0xaa7ac7e4ced68ae83a8f17de629efe920b097683	1,999,938.08	2.0221%
4	0x41e7e21c4fd8b9d0d505836a72d74d22a5303e4f	1,870,981.832059298	1.8918%
5	0x1462d1728986cf085556017999f99b23d81018c9	1,504,686.220255468	1.5214%
6	0x3dcae610c4dbef3e30cfe1b0c7ff3d1e89e5800a	1,485,504.942304282	1.5020%
7	0xcf3265f1a36f26a25f3338d74006fb97f79f680a	1,475,837.208168361	1.4922%
8	0x26324bb20ab4e48ff4155c92ebb328c6fdcf56b3	1,349,018.004399685	1.3640%
9	0xca17ec0747ae14ed62a63db6f6c022b59a5982c2	1,203,386.029268387	1.2168%
10	0xe3244c2e995efd4fc7ed28146b24b3f7ae1e336c	1,172,449.99944	1.1855%

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] IPancakeERC20
  - [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 #

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

+ [Int] IPancakeRouter01
  - [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] factory

  - [Ext] WETH
  - [Ext] quote
  - [Ext] getamountOut
  - [Ext] getamountIn
  - [Ext] getamountsOut
  - [Ext] getamountsIn

+ [Int] IPancakeRouter02 (IPancakeRouter01)
  - [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
  - [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

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

+ [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] EnumerableSet
  - [Prv] _add #
  - [Prv] _remove #
  - [Prv] _contains
  - [Prv] _length
  - [Prv] _at
  - [Int] add #
  - [Int] remove #
  - [Int] contains
  - [Int] length
  - [Int] at
  - [Int] add #
  - [Int] remove #
  - [Int] contains
  - [Int] length
  - [Int] at
  - [Int] add #
  - [Int] remove #
  - [Int] contains
  - [Int] length

- [Int] at

+ EarnDOGE (IBEP20, Ownable)
  - [Prv] _isTeam
  - [Pub] <Constructor> #
  - [Prv] _transfer #
  - [Prv] _taxedTransfer #
  - [Prv] _feelessTransfer #
  - [Prv] _calculateFee
  - [Pub] isExcludedFromStaking
  - [Pub] _getTotalShares
  - [Prv] _addToken #
  - [Prv] _removeToken #
  - [Prv] _newDividentsOf
  - [Prv] _distributeStake #
  - [Prv] claimDOGE #
  - [Prv] _swapContractToken #
    - modifiers: lockTheSwap
  - [Prv] _swapTokenForBNB #
  - [Prv] _addLiquidity #
  - [Pub] getLiquidityReleaseTimeInSeconds
  - [Pub] getBurnedTokens
  - [Pub] getLimits
  - [Pub] getTaxes
  - [Pub] getAddressSellLockTimeInSeconds
  - [Pub] getSellLockTimeInSeconds
  - [Pub] AddressResetSellLock #
  - [Pub] DOGEWithdraw #
  - [Pub] getDividents
  - [Pub] TeamWithdrawMarketingBNB #
    - modifiers: onlyOwner
  - [Pub] TeamWithdrawMarketingBNB #
    - modifiers: onlyOwner
  - [Pub] TeamSwitchManualBNBConversion #
    - modifiers: onlyOwner
  - [Pub] TeamDisableSellLock #
    - modifiers: onlyOwner
  - [Pub] TeamSetSellLockTime #
    - modifiers: onlyOwner
  - [Pub] TeamSetTaxes #
    - modifiers: onlyOwner
  - [Pub] TeamChangeMarketingShare #
    - modifiers: onlyOwner
  - [Pub] TeamCreateLPandBNB #
    - modifiers: onlyOwner
  - [Pub] TeamUpdateLimits #
    - modifiers: onlyOwner
  - [Pub] SetupEnableTrading #
    - modifiers: onlyOwner
  - [Pub] SetupLiquidityTokenAddress #
    - modifiers: onlyOwner
  - [Pub] TeamUnlockLiquidityInSeconds #
    - modifiers: onlyOwner
  - [Prv] _prolongLiquidityLock #
  - [Pub] TeamReleaseLiquidity #

  - modifiers: onlyOwner
- [Pub] TeamRemoveLiquidity #
  - modifiers: onlyOwner
- [Pub] TeamRemoveRemainingBNB #
  - modifiers: onlyOwner
- [Ext] <Fallback> ($)
- [Ext] <Fallback> ($)
- [Ext] getOwner
- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Ext] 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.	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 _getTotalShares() uses the loop to find and decrease shares from the _excludedFromStaking list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

```
function _getTotalShares() public view returns (uint256){
    uint256 shares=_circulatingSupply;
    //substracts all excluded from shares, excluded list is limited to 30
    // to avoid creating a Honeypot through OutOfGas exeption
    for(uint i=0; i<_excludedFromStaking.length(); i++){
        shares-=_balances[_excludedFromStaking.at(i)];
    }
    return shares;
}
```

Recommendation:

Check that the excluded array length is not too big.

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

- Owner can withdraw marketing balance.

```
function TeamWithdrawMarketingBNB() public onlyOwner{
    uint256 amount=marketingBalance;
    marketingBalance=0;
    (bool sent,) =TeamWallet.call{value: (amount)}("");
    require(sent,"withdraw failed");
}
```

```
function TeamWithdrawMarketingBNB(uint256 amount⬆) public onlyOwner{
    require(amount⬆<=marketingBalance);
    marketingBalance-=amount⬆;
    (bool sent,) =TeamWallet.call{value: (amount⬆)}("");
    require(sent,"withdraw failed");
}
```

- Owner can disable auto call of _swapContractToken function.

```
function TeamSwitchManualBNBConversion(bool manual⬆) public onlyOwner{
    manualConversion=manual⬆;
}
```

- Owner can disable sell lock.

```
function TeamDisableSellLock(bool disabled⬆) public onlyOwner{
    sellLockDisabled=disabled⬆;
}
```

- Owner can change sell lock time.

```
function TeamSetSellLockTime(uint256 sellLockSeconds⬆)public onlyOwner{
        require(sellLockSeconds⬆<=MaxSellLockTime,"Sell Lock time too high");
        sellLockTime=sellLockSeconds⬆;
}
```

- Owner can change taxes.

```
function TeamSetTaxes(uint8 burnTaxes⬆, uint8 liquidityTaxes⬆, uint8 stakingTaxes⬆,uint8 buyTax⬆, uint8 sellTax⬆, uint8 transferTax⬆) public onlyOwner{
    uint8 totalTax=burnTaxes⬆+liquidityTaxes⬆+stakingTaxes⬆;
    require(totalTax==100, "burn+liq+marketing needs to equal 100%");

    _burnTax=burnTaxes⬆;
    _liquidityTax=liquidityTaxes⬆;
    _stakingTax=stakingTaxes⬆;

    _buyTax=buyTax⬆;
    _sellTax=sellTax⬆;
    _transferTax=transferTax⬆;
}
```

- Owner can change marketing share (percentage of BNB that goes to marketing).

```
function TeamChangeMarketingShare(uint8 newShare⬆) public onlyOwner{
    require(newShare⬆<=50);
    marketingShare=newShare⬆;
}
```

- Owner can manually call _swapContractToken function.

```
function TeamCreateLPandBNB() public onlyOwner{
_swapContractToken();
}
```

- Owner can change balance and sell limits.

```
function TeamUpdateLimits(uint256 newBalanceLimit⬆, uint256 newSellLimit⬆) public onlyOwner{
    //SellLimit needs to be below 1% to avoid a Large Price impact when generating auto LP
    require(newSellLimit⬆<_circulatingSupply/100);
    //Adds decimals to limits
    newBalanceLimit⬆=newBalanceLimit⬆*10**_decimals;
    newSellLimit⬆=newSellLimit⬆*10**_decimals;
    //Calculates the target Limits based on supply
    uint256 targetBalanceLimit=_circulatingSupply/BalanceLimitDivider;
    uint256 targetSellLimit=_circulatingSupply/SellLimitDivider;

    require((newBalanceLimit⬆>=targetBalanceLimit),
    "newBalanceLimit needs to be at least target");
    require((newSellLimit⬆>=targetSellLimit),
    "newSellLimit needs to be at least target");

    balanceLimit = newBalanceLimit⬆;
    sellLimit = newSellLimit⬆;
}
```

- Owner can enable trading(already called).

```
function SetupEnableTrading() public onlyOwner{
    ftrace | funcSig
    tradingEnabled=true;
}
```

- Owner can change liquidity token address.

```
function SetupLiquidityTokenAddress(address liquidityTokenAddress⬆) public onlyOwner{
    ftrace | funcSig
    _liquidityTokenAddress=liquidityTokenAddress⬆;
}
```

- Owner can increase _liquidityUnlockTime.

```
function TeamUnlockLiquidityInSeconds(uint256 secondsUntilUnlock⬆) public onlyOwner{
    ftrace | funcSig
    _prolongLiquidityLock(secondsUntilUnlock⬆+block.timestamp);
}
```

- Owner can withdraw liquidity to team wallet if it is not locked.

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

    IPancakeERC20 liquidityToken = IPancakeERC20(_liquidityTokenAddress);
    uint256 amount = liquidityToken.balanceOf(address(this));

    //Liquidity release if something goes wrong at start
    liquidityToken.transfer(TeamWallet, amount);

}
```

- Owner can remove liquidity.

```
function TeamRemoveLiquidity(bool addToStaking⇑) public onlyOwner{
    ftrace | funcSig
    //Only callable if liquidity Unlock time is over
    require(block.timestamp >= _liquidityUnlockTime, "Not yet unlocked");
    _liquidityUnlockTime=block.timestamp+DefaultLiquidityLockTime;
    IPancakeERC20 liquidityToken = IPancakeERC20(_liquidityTokenAddress);
    uint256 amount = liquidityToken.balanceOf(address(this));

    liquidityToken.approve(address(_pancakeRouter),amount);
    //Removes Liquidity and either distributes liquidity BNB to stakers, or
    // adds them to marketing Balance
    //Token will be converted
    //to Liquidity and Staking BNB again
    uint256 initialBNBBalance = address(this).balance;
    _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(
        address(this),
        amount,
        0,
        0,
        address(this),
        block.timestamp
        );
    uint256 newBNBBalance = address(this).balance-initialBNBBalance;
    if(addToStaking⇑){
        _distributeStake(newBNBBalance);
    }
    else{
        marketingBalance+=newBNBBalance;
    }

}
```

- **Owner can withdraw contract balance if it is not locked.**

```
function TeamRemoveRemainingBNB() public onlyOwner{
    ftrace | funcSig
    require(block.timestamp >= _liquidityUnlockTime, "Not yet unlocked");
    _liquidityUnlockTime=block.timestamp+DefaultLiquidityLockTime;
    (bool sent,) =TeamWallet.call{value: (address(this).balance)}("");
    require(sent);
}
```

Conclusion

Smart contracts contain low severity issues! 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.