Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

Olympia

Deployer address

0x5192ac0F1718E9CE5777F77A87F6a71a40BDd2De

Client contacts:

Olympia team

Blockchain

Binance Smart Chain

Project website:

https://sphereofolympia.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 Olympia to perform an audit of smart contracts:

https://bscscan.com/address/0x79395a77c8ddbefb74f242242960cac7368fd64e#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

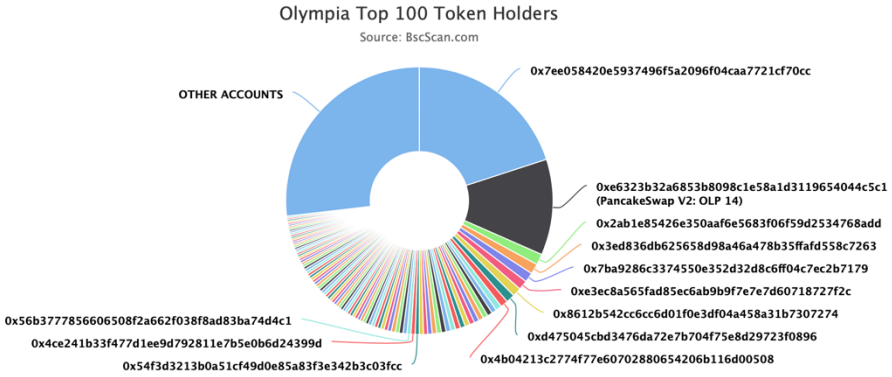
Token contract details for 07.11.2021

Contract name	Olympia
Contract address	0x79395a77c8ddBEFb74f242242960CAc7368fD64E
Total supply	100,000,000,000
Token ticker	OLP
Decimals	18
Token holders	1,111
Transactions count	9,483
Top 100 holders dominance	73.16%
Dividend tracker	0x3d5aEd09FE43f369404136D9281D9F83e209eEf2
Marketing wallet	0x0A8B049ad6f89CE923365C74A2BAA34532eF4d95
BNB rewards fee	3
Uniswap V2 pair	0xE6323B32A6853b8098c1E58A1d3119654044c5c1
Contract deployer address	0x5192ac0F1718E9CE5777F77A87F6a71a40BDd2De
Contract's current owner address	0x591b71270958Fa020F4E698c2cBf76897763175d

Olympia Token Distribution

The top 100 holders collectively own 73.16% (73,160,380,082.62 Tokens) of Olympia

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 1,111

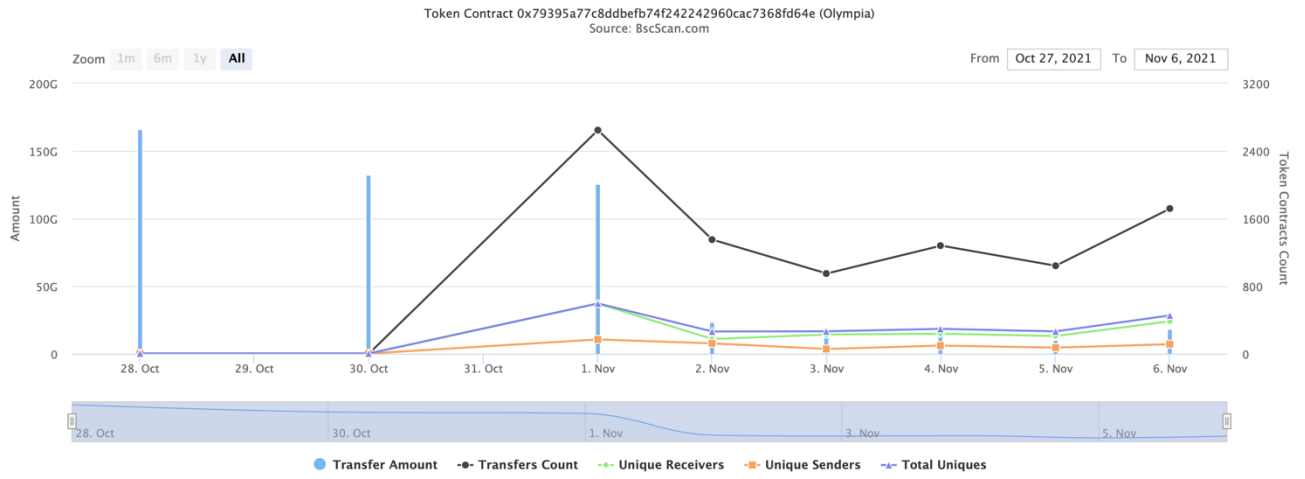


(A total of 73,160,380,082.62 tokens held by the top 100 accounts from the total supply of 100,000,000,000.00 token)

Olympia Contract Interaction Details

Time Series: Token Contract Overview

Thu 28, Oct 2021 - Sat 6, Nov 2021



Olympia Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0x7ee058420e5937496f5a2096f04caa7721cf70cc	20,000,000,000	20.0000%
2	PancakeSwap V2: OLP 14	11,594,987,791.89682550005461371	11.5950%
3	0x2ab1e85426e350aaf6e5683f06f59d2534768add	1,319,421,224.271968340988845645	1.3194%
4	0x3ed836db625658d98a46a478b35ffafd558c7263	1,237,846,793.0625	1.2378%
5	0x7ba9286c3374550e352d32d8c6ff04c7ec2b7179	1,154,223,976	1.1542%
6	0xe3ec8a565fad85ec6ab9b9f7e7e7d60718727f2c	1,148,748,750	1.1487%
7	0x8612b542cc6cc6d01f0e3df04a458a31b7307274	1,080,000,000	1.0800%
8	0xd475045cbd3476da72e7b704f75e8d29723f0896	1,061,593,571.483434013024289125	1.0616%
9	0x4b04213c2774f77e60702880654206b116d00508	900,000,000	0.9000%
10	0x8cd6127c34b3ac7749430ae466a0e8f9ebe0d364	753,115,500	0.7531%

Contract functions details

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

+ [Int] IERC20Metadata (IERC20)
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals

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

+ [Int] DividendPayingTokenInterface
  - [Ext] dividendOf
  - [Ext] distributeDividends ($)
  - [Ext] withdrawDividend #

+ [Int] DividendPayingTokenOptionalInterface
  - [Ext] withdrawableDividendOf
  - [Ext] withdrawnDividendOf
  - [Ext] accumulativeDividendOf

+ ERC20 (Context, IERC20, IERC20Metadata)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #
  - [Int] _approve #
  - [Int] _beforeTokenTransfer #

+ [Lib] IterableMapping
  - [Pub] get
  - [Pub] getIndexOfKey
  - [Pub] getKeyAtIndex
  - [Pub] size

  - [Pub] set #
  - [Pub] remove #

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

+ [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] 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 #

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

+ OLPTOKEN (ERC20, Ownable)
- [Pub] <Constructor> #
  - modifiers: ERC20
- [Ext] <Fallback> ($)
- [Pub] updateDividendTracker #
  - modifiers: onlyOwner
- [Pub] updateUniswapV2Router #
  - modifiers: onlyOwner
- [Pub] excludeFromFees #
  - modifiers: onlyOwner
- [Pub] excludeMultipleAccountsFromFees #
  - modifiers: onlyOwner
- [Ext] setMarketingWallet #
  - modifiers: onlyOwner
- [Ext] setBNBRewardsFee #
  - modifiers: onlyOwner
- [Ext] setLiquiditFee #
  - modifiers: onlyOwner
- [Ext] setMarketingFee #
  - modifiers: onlyOwner
- [Pub] setAutomatedMarketMakerPair #
  - modifiers: onlyOwner
- [Ext] blacklistAddress #
  - modifiers: onlyOwner
- [Prv] _setAutomatedMarketMakerPair #
- [Pub] updateGasForProcessing #
  - modifiers: onlyOwner
- [Ext] updateClaimWait #
  - modifiers: onlyOwner
- [Ext] getClaimWait
- [Ext] getTotalDividendsDistributed
- [Pub] isExcludedFromFees

- [Pub] withdrawableDividendOf
- [Pub] dividendTokenBalanceOf
- [Ext] excludeFromDividends #
  - modifiers: onlyOwner
- [Ext] setmaxTransactionPercentage #
  - modifiers: onlyOwner
- [Ext] setmaxTokensPerAddressPercentage #
  - modifiers: onlyOwner
- [Ext] setIsTxLimitExempt #
  - modifiers: onlyOwner
- [Pub] setIsExcludedFromAntiWhale #
  - modifiers: onlyOwner
- [Pub] setIsExcludedFromTransactionCooldown #
  - modifiers: onlyOwner
- [Pub] setIsPremium #
  - modifiers: onlyOwner
- [Pub] deletePremiumUserByAddress #
  - modifiers: onlyOwner
- [Ext] getPremiumList
- [Pub] setTransactionCooldownTime #
  - modifiers: onlyOwner
- [Pub] setBuyFees #
  - modifiers: onlyOwner
- [Pub] setSellFees #
  - modifiers: onlyOwner
- [Prv] setFeeOnBuy #
- [Prv] setFeeOnSell #
- [Ext] getAccountDividendsInfo
- [Ext] getAccountDividendsInfoAtIndex
- [Ext] processDividendTracker #
- [Ext] claim #
- [Ext] getLastProcessedIndex
- [Ext] getNumberOfDividendTokenHolders
- [Int] _transfer #
- [Prv] swapAndSendToFee #
- [Prv] swapAndLiquify #
- [Prv] swapTokensForEth #
- [Prv] addLiquidity #
- [Prv] swapAndSendDividends #

+ DividendPayingToken (ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface)
  - [Pub] <Constructor> #
    - modifiers: ERC20
  - [Ext] <Fallback> ($)
  - [Pub] distributeDividends ($)
  - [Pub] withdrawDividend #
  - [Int] _withdrawDividendOfUser #
  - [Pub] dividendOf
  - [Pub] withdrawableDividendOf
  - [Pub] withdrawnDividendOf
  - [Pub] accumulativeDividendOf
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #

  - [Int] _setBalance #

+ OLPDividendTracker (Ownable, DividendPayingToken)
  - [Pub] <Constructor> #
    - modifiers: DividendPayingToken
  - [Int] _transfer #
  - [Pub] withdrawDividend #
  - [Ext] excludeFromDividends #
    - modifiers: onlyOwner
  - [Ext] updateClaimWait #
    - modifiers: onlyOwner
  - [Ext] getLastProcessedIndex
  - [Ext] getNumberOfTokenHolders
  - [Ext] setMinimumTokenBalanceForDividends #
    - modifiers: onlyOwner
  - [Pub] getAccount
  - [Pub] getAccountAtIndex
  - [Prv] canAutoClaim
  - [Ext] setBalance #
    - modifiers: onlyOwner
  - [Pub] process #
  - [Pub] processAccount #
    - modifiers: onlyOwner

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

+ [Lib] SafeMathInt
  - [Int] mul
  - [Int] div
  - [Int] sub
  - [Int] add
  - [Int] abs
  - [Int] toUint256Safe

+ [Lib] SafeMathUint
  - [Int] toInt256Safe

($) = 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 `excludeMultipleAccountsFromFees()` uses the loop to exclude multiple accounts from fees. Function will be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function excludeMultipleAccountsFromFees(address[] memory accounts, bool excluded) public onlyOwner {
    for(uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

- The function `deletePremiumUserByAddress()` uses the loop to delete account from premium list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long addresses list.

```
function deletePremiumUserByAddress(address account) public onlyOwner {
    for(uint i = 0; i < listPremium.length; i++) {
        if(listPremium[i] == account) {
            isPremium[account].premium = false;
            isPremium[account].balance = 0;
            delete listPremium[i];
            emit DeletePremiumUserFromList(account);
        }
    }
}
```

Recommendation:
Be careful about accounts array length.

Notes:

- Owner can change dividend tracker to not audited and some functions may work in different ways.

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

- Owner can change dividend tracker.

```
function updateDividendTracker(address newAddress) public onlyOwner {
    require(newAddress != address(dividendTracker), "OLP: The dividend tracker already has that address");

    OLPDividendTracker newDividendTracker = OLPDividendTracker(payable(newAddress));

    require(newDividendTracker.owner() == address(this), "OLP: The new dividend tracker must be owned by the OLP token contract");

    newDividendTracker.excludeFromDividends(address(newDividendTracker));
    newDividendTracker.excludeFromDividends(address(this));
    newDividendTracker.excludeFromDividends(owner());
    newDividendTracker.excludeFromDividends(address(uniswapV2Router));

    emit UpdateDividendTracker(newAddress, address(dividendTracker));

    dividendTracker = newDividendTracker;
}
```

- Owner can change Uniswap router address.

```
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(newAddress != address(uniswapV2Router), "OLP: The router already has that address");
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

- Owner can include in and exclude from the fees.

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "OLP: Account is already the value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}

function excludeMultipleAccountsFromFees(address[] memory accounts, bool excluded) public onlyOwner {
    for(uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

- Owner can change marketing wallet address.

```
function setMarketingWallet(address payable wallet) external onlyOwner{
    _marketingWalletAddress = wallet;
}
```

- Owner can exclude and include addresses in automatedMarketMakerPairs array.

```
function setAutomatedMarketMakerPair(address pair, bool value) public onlyOwner {
    require(pair != uniswapV2Pair, "OLP: The PancakeSwap pair cannot be removed from automatedMarketMakerPairs");

    _setAutomatedMarketMakerPair(pair, value);
}
```

- Owner can change BNB rewards, liquidity and marketing fees.

```
function setBNBRewardsFee(uint256 value) external onlyOwner{
    BNBRewardsFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);
}

function setLiquiditFee(uint256 value) external onlyOwner{
    liquidityFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);
}

function setMarketingFee(uint256 value) external onlyOwner{
    marketingFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee);

}
```

- Owner can add / remove addresses from blacklist.

```
function blacklistAddress(address account, bool value) external onlyOwner{
    _isBlacklisted[account] = value;
}
```

- Owner can change gas for processing.

```
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue >= 200000 && newValue <= 500000, "OLP: gasForProcessing must be between 200,000 and 500,000");
    require(newValue != gasForProcessing, "OLP: Cannot update gasForProcessing to same value");
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

- Owner can update claimWait value.

```
function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

- Owner can exclude from dividends.

```
function excludeFromDividends(address account) external onlyOwner{
    dividendTracker.excludeFromDividends(account);
}
```

- Owner can change maximum transaction amount.

```
function setmaxTransactionPercentage(uint256 _percentage) external onlyOwner {
    maxTxAmount = _totalSupply.mul(_percentage).div(1000);
}
```

- Owner can change maximum token amount per wallet.

```
function setmaxTokensPerAddressPercentage(uint256 _percentage) external onlyOwner {
    maxTokensPerAddress = _totalSupply.mul(_percentage).div(1000);
}
```

- **Owner can manage addresses in isTxLimitExempt, isExcludedFromAntiWhale and isExcludedFromTransactionlock arrays.**

```
function setIsTxLimitExempt(address holder, bool exempt) external onlyOwner {
    isTxLimitExempt[holder] = exempt;
}

function setIsExcludedFromAntiWhale(address account, bool excluded) public onlyOwner {
    isExcludedFromAntiWhale[account] = excluded;
}

function setIsExcludedFromTransactionCooldown(address account, bool excluded) public onlyOwner {
    isExcludedFromTransactionlock[account] = excluded;
}
```

- **Owner can add / remove addresses from premium list.**

```
function setIsPremium(address account, uint balance) public onlyOwner {
    require(isPremium[account].premium == false, "User must not be premium before to avoid duplicate");
    isPremium[account].premium = true;
    isPremium[account].balance = balance;

    //update premium list
    listPremium.push(account);
}

function deletePremiumUserByAddress(address account) public onlyOwner {
    for(uint i = 0; i < listPremium.length; i++) {
        if(listPremium[i] == account) {
            isPremium[account].premium = false;
            isPremium[account].balance = 0;
            delete listPremium[i];
            emit DeletePremiumUserFromList(account);
        }
    }
}
```

- **Owner can change transactionLockTime value.**

```
function setTransactionCooldownTime(uint256 transactiontime) public onlyOwner {
    transactionLockTime = transactiontime;
}
```

- **Owner can change buy and sell fees.**

```
function setBuyFees(uint256 reflectFee, uint256 marketFee, uint256 liquidFee) public onlyOwner {
    _buyFees._reflectFee = reflectFee;
    _buyFees._marketFee = marketFee;
    _buyFees._liquidFee = liquidFee;
}

function setSellFees(uint256 reflectFee, uint256 marketFee, uint256 liquidFee) public onlyOwner {
    _sellFees._reflectFee = reflectFee;
    _sellFees._marketFee = marketFee;
    _sellFees._liquidFee = liquidFee;
}
```

Conclusion

Smart contracts contain low severity issues and owner privileges! Liquidity pair contract's security is not checked due to out of scope. The further transfers and operations with the funds raise are not related to this particular contract.

Liquidity locking details provided by the team:
https://www.pinksale.finance/#/pinklock/detail/0x79395a77c8ddBEFb74f242242960CAc7368fD64E?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.