Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

August, 2021

Audit Details

Audited project

DAIKOKUTEN SAMA

Deployer address

0x2537260d7e11cb08E6028453713B2e958Fab2E2D

Client contacts:

DAIKOKUTEN SAMA team

Blockchain

Binance Smart Chain

Project website:

https://daikokuten.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 DAIKOKUTEN SAMA to perform an audit of smart contracts:
https://bscscan.com/address/0x1143Ed6C433751772c398Be05158D6d22484B047#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

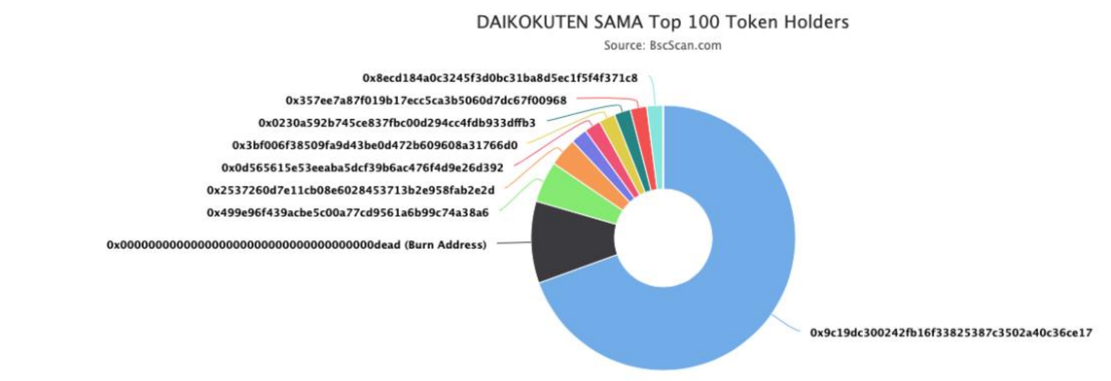
Token contract details for 08.08.2021

Contract name	DKKS
Contract address	0x1143Ed6C433751772c398Be05158D6d22484B047
Total supply	1,000,000,000,000,000
Token ticker	DKKS
Decimals	9
Token holders	10
Transactions count	10
Top 100 holders dominance	100.00%
Liquidity fee	0
Tax fee	0
Total fees	0
Uniswap V2 pair	0x0bda517423ea48c6c3bd7e9f8123b9dc8d726aa7
Contract deployer address	0x2537260d7e11cb08E6028453713B2e958Fab2E2D
Contract's current owner address	0x2537260d7e11cb08e6028453713b2e958fab2e2d

DAIKOKUTEN SAMA Token Distribution

The top 100 holders collectively own 100.00% (1,000,000,000,000,000.00 Tokens) of DAIKOKUTEN SAMA

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 10

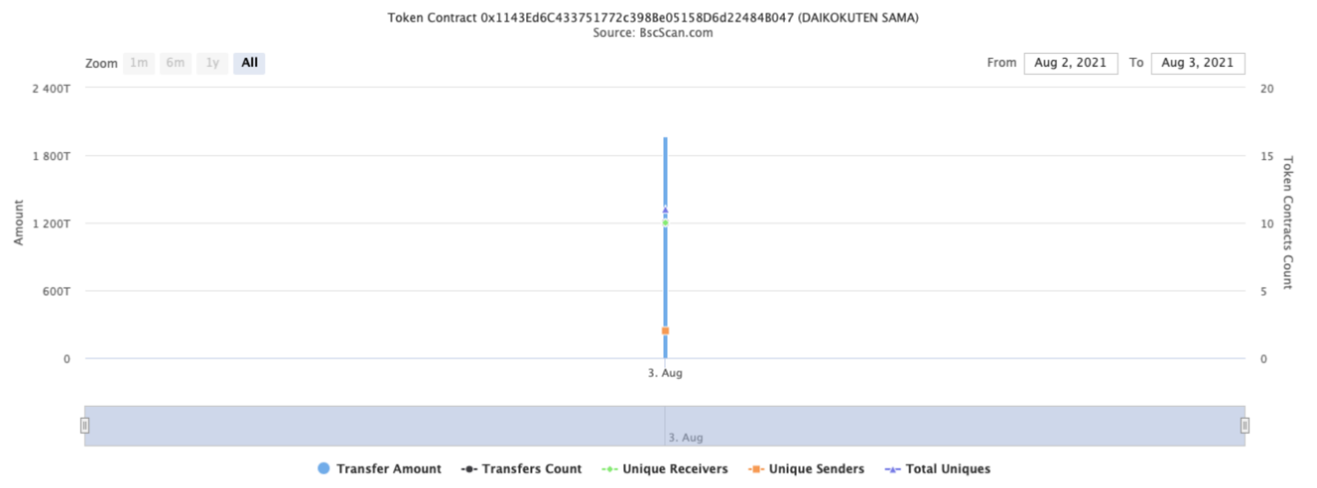


(A total of 1,000,000,000,000,000.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

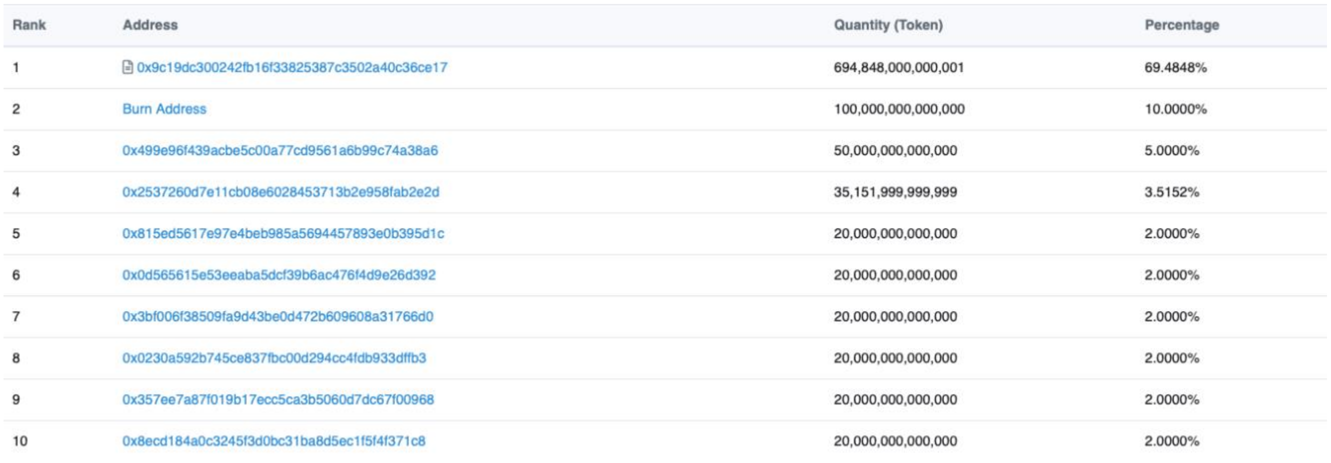
DAIKOKUTEN SAMA Contract Interaction Details

Time Series: Token Contract Overview

Tue 3, Aug 2021 - Tue 3, Aug 2021



DAIKOKUTEN SAMA Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	0x9c19dc300242fb16f33825387c3502a40c36ce17	694,848,000,000,001	69.4848%
2	Burn Address	100,000,000,000,000	10.0000%
3	0x499e96f439acbe5c00a77cd9561a6b99c74a38a6	50,000,000,000,000	5.0000%
4	0x2537260d7e11cb08e6028453713b2e958fab2e2d	35,151,999,999,999	3.5152%
5	0x815ed5617e97e4beb985a5694457893e0b395d1c	20,000,000,000,000	2.0000%
6	0x0d565615e53eeaba5dcf39b6ac476f4d9e26d392	20,000,000,000,000	2.0000%
7	0x3bf006f38509fa9d43be0d472b609608a31766d0	20,000,000,000,000	2.0000%
8	0x0230a592b745ce837fbc00d294cc4fdb933dffb3	20,000,000,000,000	2.0000%
9	0x357ee7a87f019b17ecc5ca3b5060d7dc67f00968	20,000,000,000,000	2.0000%
10	0x8ecd184a0c3245f3d0bc31ba8d5ec1f5f4f371c8	20,000,000,000,000	2.0000%

Contract functions details

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

 + [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

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

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

+ DKKS (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] minimumTokensBeforeSwapAmount
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] swapTokens #
    - modifiers: lockTheSwap
  - [Prv] swapTokensForEth #
  - [Prv] swapETHForTokens #
  - [Prv] addLiquidity #
  - [Prv] _tokenTransfer #
  - [Prv] _transferStandard #
  - [Prv] _transferToExcluded #
  - [Prv] _transferFromExcluded #
  - [Prv] _transferBothExcluded #
  - [Prv] _reflectFee #
  - [Prv] _getValues
  - [Prv] _getTValues
  - [Prv] _getRValues
  - [Prv] _getRate
  - [Prv] _getCurrentSupply
  - [Prv] _takeLiquidity #
  - [Prv] calculateTaxFee
  - [Prv] calculateLiquidityFee
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Pub] isExcludedFromFee
  - [Pub] excludeFromFee #
    - modifiers: onlyOwner
  - [Pub] includeInFee #
    - modifiers: onlyOwner

- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setMarketingDivisor #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToLiquidity #
  - modifiers: onlyOwner
- [Ext] setMaxTokenHolder #
  - modifiers: onlyOwner
- [Ext] setMarketingAddress #
  - modifiers: onlyOwner
- [Pub] changeRouterVersion #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Ext] prepareForPreSale #
  - modifiers: onlyOwner
- [Ext] goLive #
  - modifiers: onlyOwner
- [Pub] transferBatch #
- [Prv] transferToAddressETH #
- [Ext] <Fallback> ($)

($) = 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

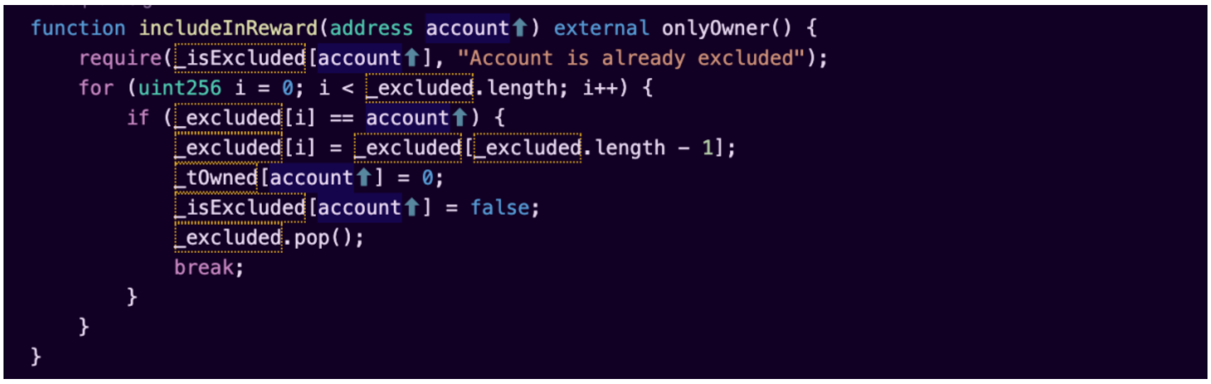
No medium severity issues found.

Low Severity Issues

1. Out of gas

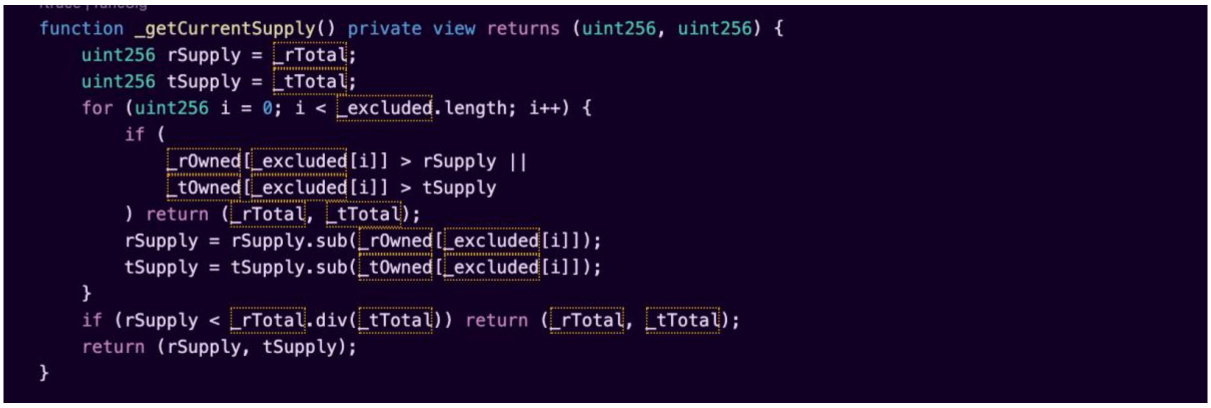
Issue:

- 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.

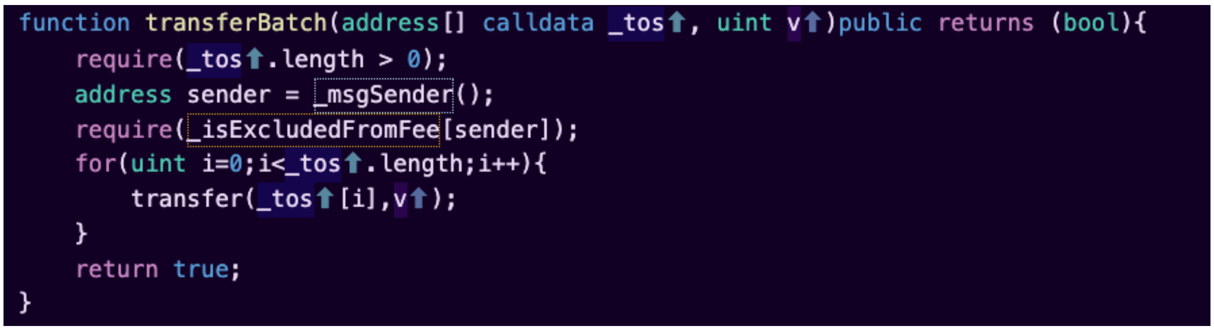


```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
      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

- The function transferBatch() uses the loop to distribute uint v amount of tokens to addresses from _tos list. It also could be aborted with OUT_OF_GAS exception if there will be a long addresses list.



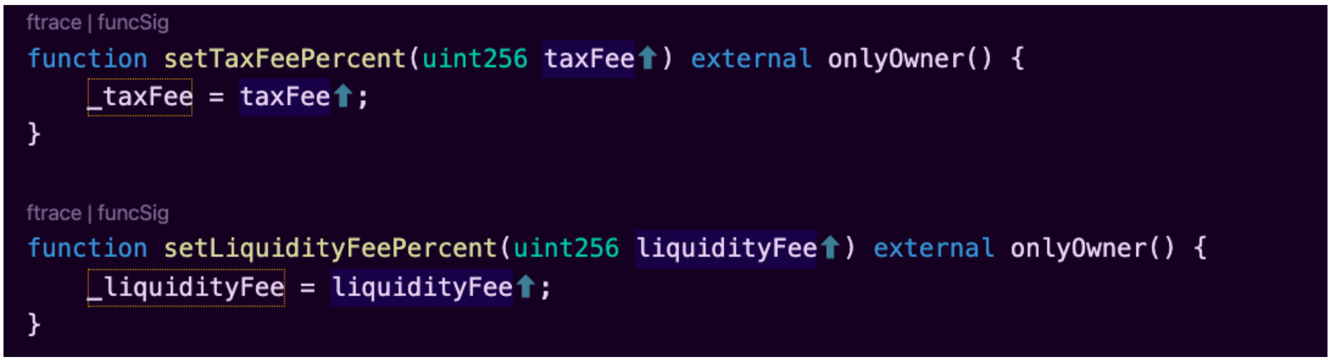
```
function transferBatch(address[] calldata _tos, uint v)public returns (bool){
    require(_tos.length > 0);
    address sender = _msgSender();
    require(_isExcludedFromFee[sender]);
    for(uint i=0;i<_tos.length;i++){
        transfer(_tos[i],v);
    }
    return true;
}
```

Recommendation:

Check that the addresses array length is not too big.

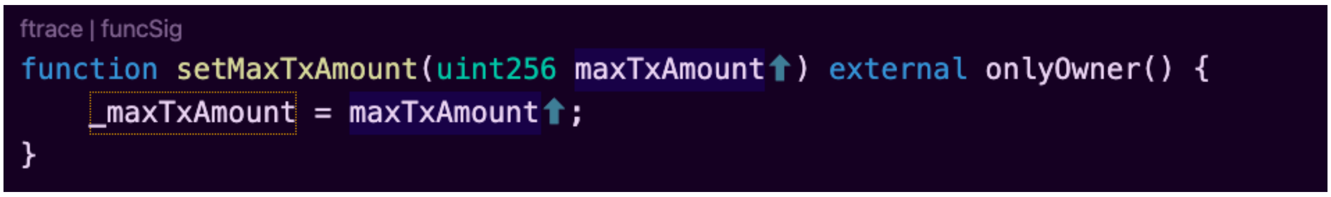
Owner privileges (In the period when the owner is not renounced)

- Owner can change tax and liquidity fees.



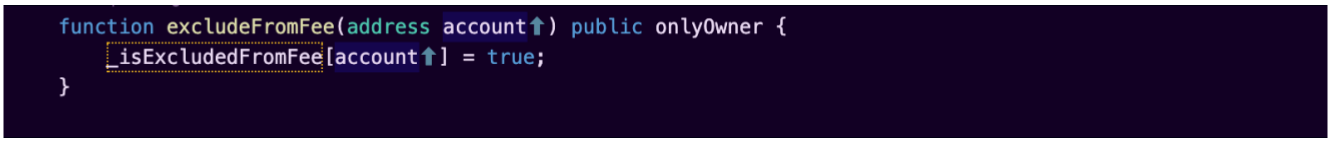
```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
```

- Owner can change maximum transaction amount.



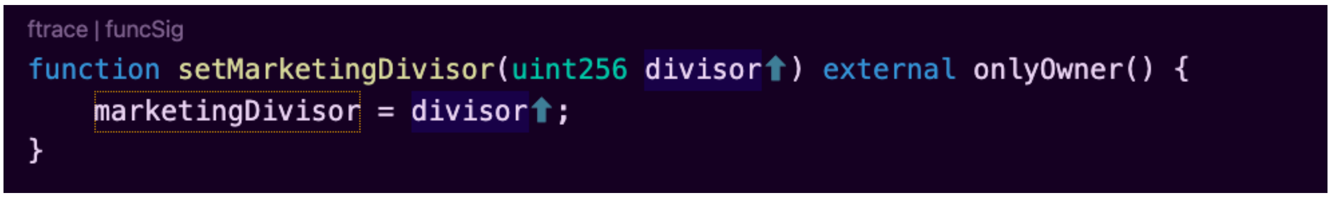
```
ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}
```

- Owner can exclude from the fee.



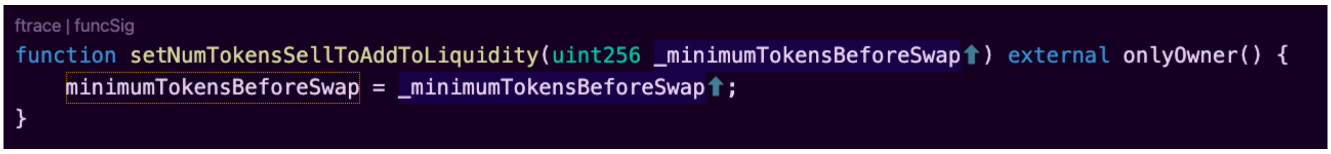
```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}
```

- Owner can change marketingDivisor.



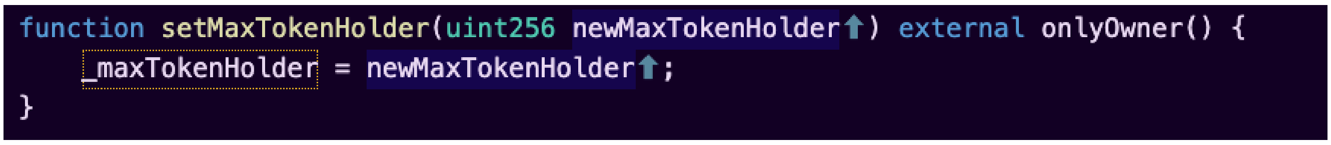
```
ftrace | funcSig
function setMarketingDivisor(uint256 divisor) external onlyOwner() {
    marketingDivisor = divisor;
}
```

- Owner can change minimum number of tokens to add to liquidity.



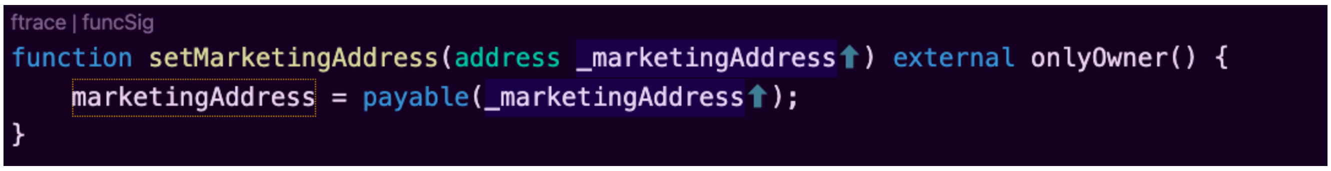
```
ftrace | funcSig
function setNumTokensSellToAddToLiquidity(uint256 _minimumTokensBeforeSwap) external onlyOwner() {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}
```

- Owner can change _maxTokenHolder value.



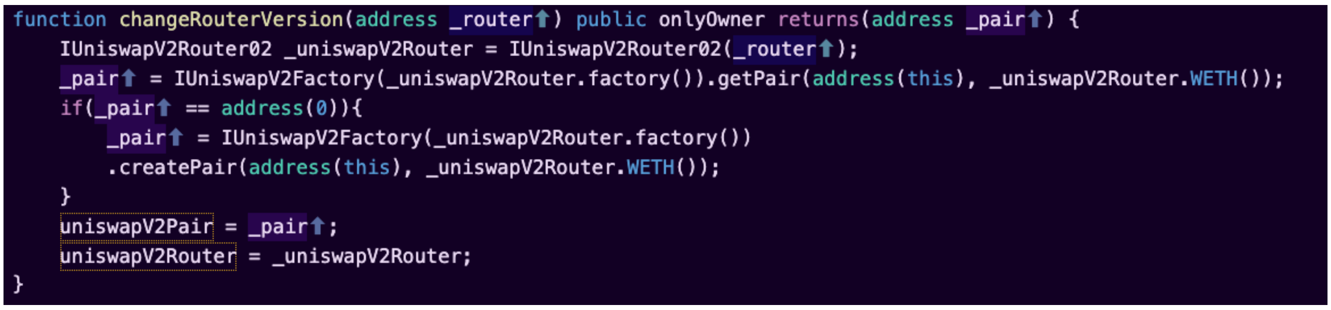
```
function setMaxTokenHolder(uint256 newMaxTokenHolder) external onlyOwner() {
    _maxTokenHolder = newMaxTokenHolder;
}
```

- Owner can change marketing address.



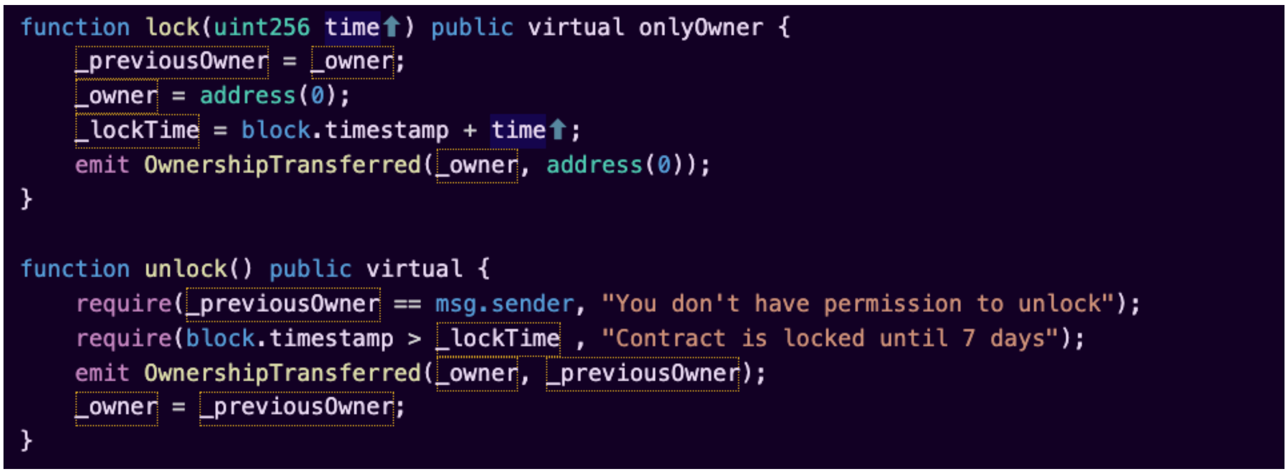
```
ftrace | funcSig
function setMarketingAddress(address _marketingAddress) external onlyOwner() {
    marketingAddress = payable(_marketingAddress);
}
```

- Owner can change router address.



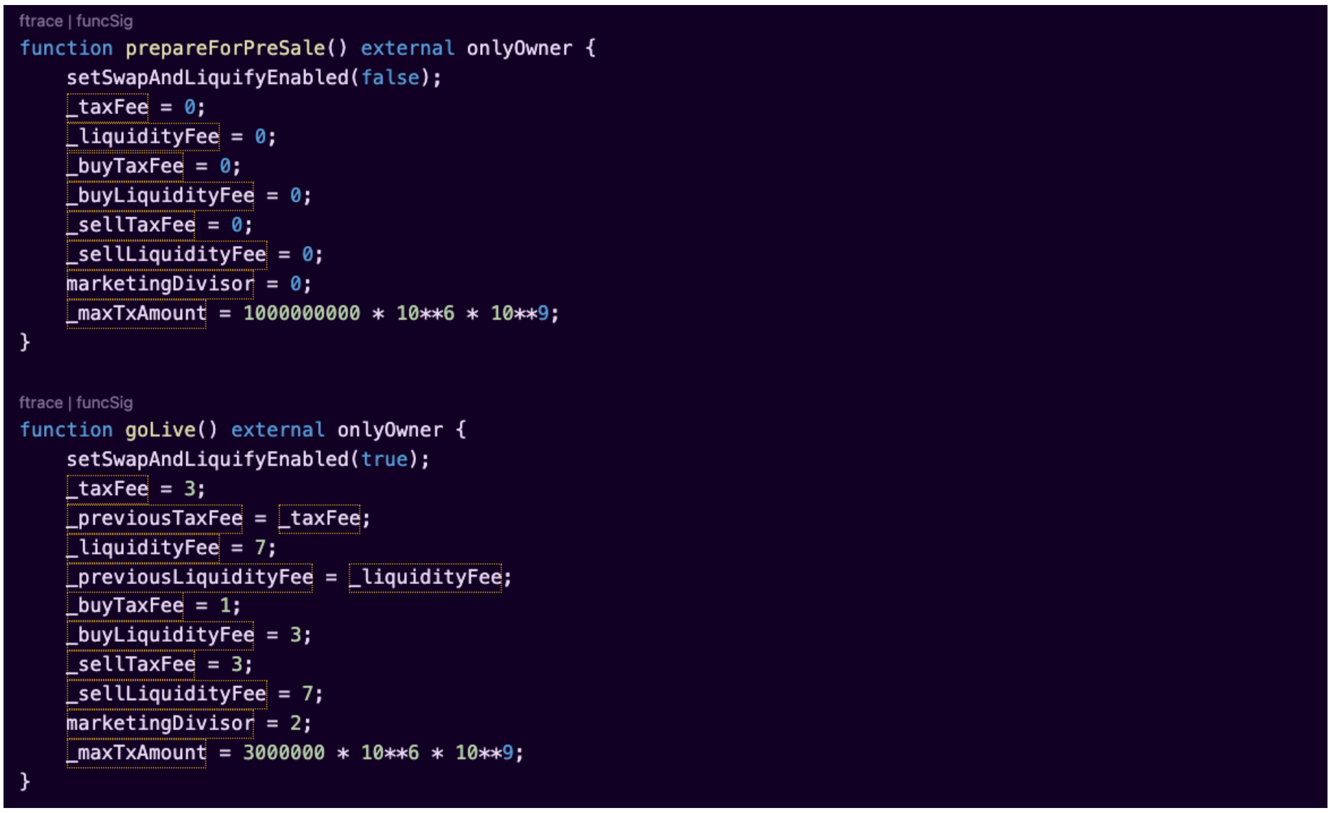
```
function changeRouterVersion(address _router) public onlyOwner returns(address _pair) {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(_router);
    _pair = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(address(this), _uniswapV2Router.WETH());
    if(_pair == address(0)){
        _pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
    }
    uniswapV2Pair = _pair;
    uniswapV2Router = _uniswapV2Router;
}
```

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



```
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}

function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

- **Owner can enable presale and live setting presets.**



```
ftrace | funcSig
function prepareForPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(false);
    _taxFee = 0;
    _liquidityFee = 0;
    _buyTaxFee = 0;
    _buyLiquidityFee = 0;
    _sellTaxFee = 0;
    _sellLiquidityFee = 0;
    marketingDivisor = 0;
    _maxTxAmount = 1000000000 * 10**6 * 10**9;
}

ftrace | funcSig
function goLive() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _taxFee = 3;
    _previousTaxFee = _taxFee;
    _liquidityFee = 7;
    _previousLiquidityFee = _liquidityFee;
    _buyTaxFee = 1;
    _buyLiquidityFee = 3;
    _sellTaxFee = 3;
    _sellLiquidityFee = 7;
    marketingDivisor = 2;
    _maxTxAmount = 3000000 * 10**6 * 10**9;
}
```

Conclusion

Smart contracts contain low severity issues! 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 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.