Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

SPIKE INU

Deployer address

0xB62495f0e828F3b515bAc313e69BB7CB38E60dDb

Client contacts:

SPIKE INU team

Blockchain

Ethereum

Project website:

https://spikeinu.io/

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.

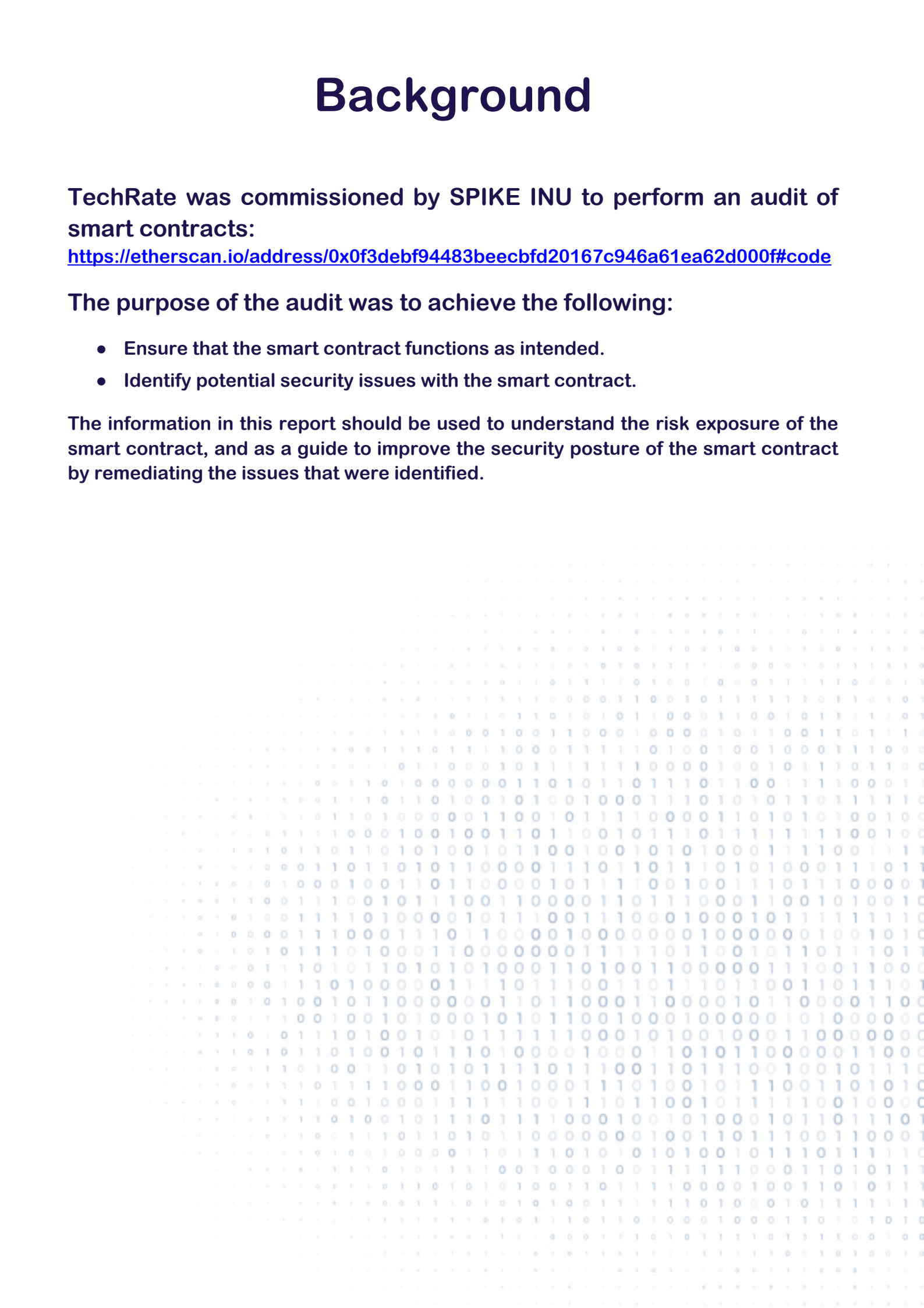
Background

TechRate was commissioned by SPIKE INU to perform an audit of smart contracts:
https://etherscan.io/address/0x0f3debf94483beecbfd20167c946a61ea62d000f#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

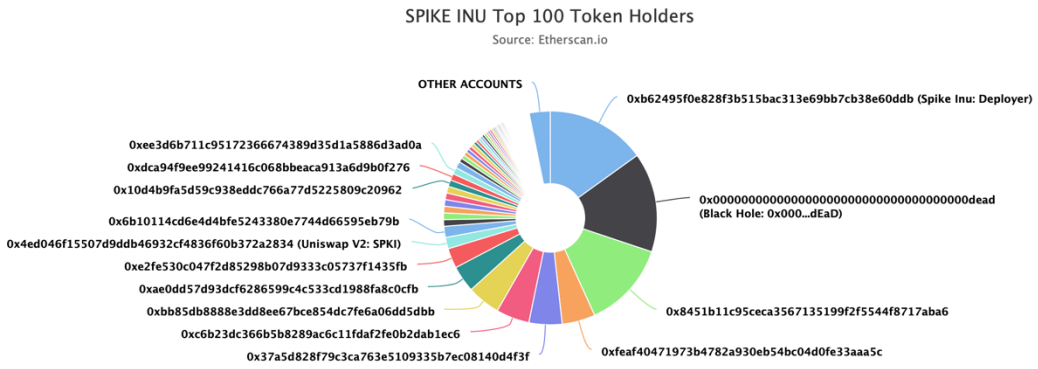
Token contract details for 05.11.2021

Contract name	SPIKE INU
Contract address	0x0f3dEBf94483beEcBFD20167C946A61Ea62d000f
Total supply	1,000,000,000,000,000
Token ticker	SPKI
Decimals	9
Token holders	674
Transactions count	1,227
Top 100 holders dominance	96.81%
Liquidity fee	4
Tax fee	1
Sell liquidity fee	6
Sell tax fee	3
Total fees	5,030,123,294,247.094036544
Uniswap V2 pair	0x4eD046f15507D9DDb46932CF4836F60b372a2834
Contract deployer address	0xB62495f0e828F3b515bAc313e69BB7CB38E60dDb
Contract's current owner address	0xB62495f0e828F3b515bAc313e69BB7CB38E60dDb

SPIKE INU Token Distribution

The top 100 holders collectively own 96.81% (968,127,207,136,592.00 Tokens) of SPIKE INU

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 691

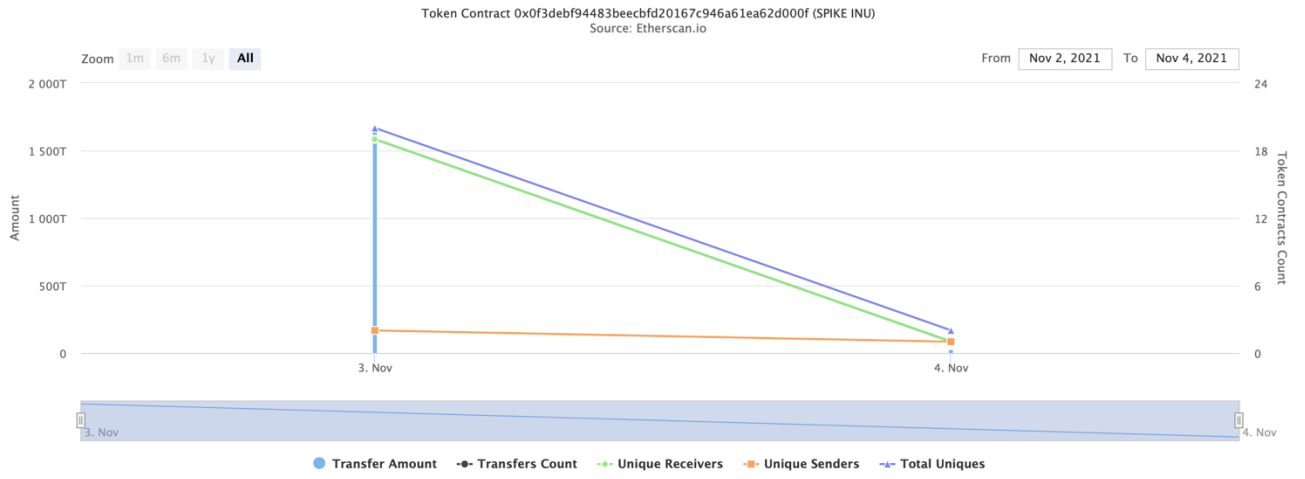


(A total of 968,127,207,136,592.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

SPIKE INU Contract Interaction Details

Time Series: Token Contract Overview

Wed 3, Nov 2021 - Thu 4, Nov 2021



SPIKE INU Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	Spike Inu: Deployer	150,743,830,132,173.816889667	15.0744%
2	Black Hole: 0x000...dEaD	150,685,397,002,090.070775936	15.0685%
3	0x8451b11c95ceca3567135199f2f5544f8717aba6	130,648,149,460,661.806272975	13.0648%
4	0xfeaf40471973b4782a930eb54bc04d0fe33aaa5c	50,249,288,254,100.694720375	5.0249%
5	0x37a5d828f79c3ca763e5109335b7ec08140d4f3f	50,249,288,254,100.694720375	5.0249%
6	0xc6b23dc366b5b8289ac6c11fdaf2fe0b2dab1ec6	50,239,169,385,892.508155174	5.0239%
7	0xbb85db8888e3dd8ee67bce854dc7fe6a06dd5dbb	50,225,851,281,411.372433642	5.0226%
8	0xae0dd57d93dcf6286599c4c533cd1988fa8c0cfb	40,201,707,118,922.009130946	4.0202%
9	0xe2fe530c047f2d85298b07d9333c05737f1435fb	30,149,622,723,789.053718363	3.0150%
10	Uniswap V2: SPKI	17,238,205,171,533.500754537	1.7238%

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 #

+ SPKI (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] getStartTime
  - modifiers: onlyOwner
- [Pub] isLockedWallet
  - modifiers: onlyOwner
- [Pub] deliver #
- [Pub] reflectionFromToken
- [Pub] tokenFromReflection
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Ext] includeInReward #
  - modifiers: onlyOwner
- [Prv] _approve #
- [Prv] killBot #
- [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] setBuyTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setSellTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setBuyLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setSellLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setMarketingDivisor #
  - modifiers: onlyOwner
- [Ext] setCharityDivisor #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToLiquidity #
  - modifiers: onlyOwner
- [Ext] setMaxTokenHolder #
  - modifiers: onlyOwner
- [Ext] setStartTime #
  - modifiers: onlyOwner
- [Ext] unlockBlackList #
  - 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.	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 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 include in and exclude from reward.

```
function excludeFromReward(address account) public onlyOwner() {

    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    require(_excluded.length < 50, "Excluded list too big");
    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;
        }
    }
}
```

- Owner can include in and exclude from the fee.

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

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

- Owner can change buy tax, sell tax, buy liquidity and sell liquidity fees.

```
function setBuyTaxFeePercent(uint256 buyTaxFee) external onlyOwner() {
    _buyTaxFee = buyTaxFee;
}

function setSellTaxFeePercent(uint256 sellTaxFee) external onlyOwner() {
    _sellTaxFee = sellTaxFee;
}

function setBuyLiquidityFeePercent(uint256 buyLiquidityFee) external onlyOwner() {
    _buyLiquidityFee = buyLiquidityFee;
}

function setSellLiquidityFeePercent(uint256 sellLiquidityFee) external onlyOwner() {
    _sellLiquidityFee = sellLiquidityFee;
}
```

- Owner can change the maximum transaction amount.

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

- Owner can change marketing and charity divisors.

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

function setCharityDivisor(uint256 divisor) external onlyOwner() {
    charityDivisor = divisor;
}
```

- Owner can change numTokensSellToAddToLiquidity value.

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

- Owner can change the maximum token amount per wallet.

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

- Owner can change start time.

```
function setStartTime(uint256 startTime) external onlyOwner() {
    _startTime = startTime;
}
```

- Owner can unlock black listed addresses.

```
function unlockBlackList(address wallet) external onlyOwner() {
    _isBotList[wallet] = false;
}
```

- Owner can change marketing address.

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

- Owner can change UniswapV2Router and UniswapV2Pair.

```
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 enable / disable swap and liquify.

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

- Owner can enable before and after presale modes.

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

function goLive() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _taxFee = 3;
    _previousTaxFee = _taxFee;
    _liquidityFee = 6;
    _previousLiquidityFee = _liquidityFee;
    _buyTaxFee = 1;
    _buyLiquidityFee = 4;
    _sellTaxFee = 3;
    _sellLiquidityFee = 6;
    marketingDivisor = 1;
    charityDivisor= 1;
    _maxTxAmount = 3000000 * 10**6 * 10**9;
}
```

Conclusion

Smart contracts do not contain 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.