Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

July, 2021

Audit Details

Audited project

BURNX 2.0

Deployer address

0x40Fa8dF858C801Cf0258121d9977d4292810267A

Client contacts:

BURNX 2.0 team

Blockchain

Ethereum

Project website:

https://burnx.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 BURNX 2.0 to perform an audit of smart contracts:
https://etherscan.io/address/0x1e950af2f6f8505c09f0ca42c4b38f10979cb22e#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.

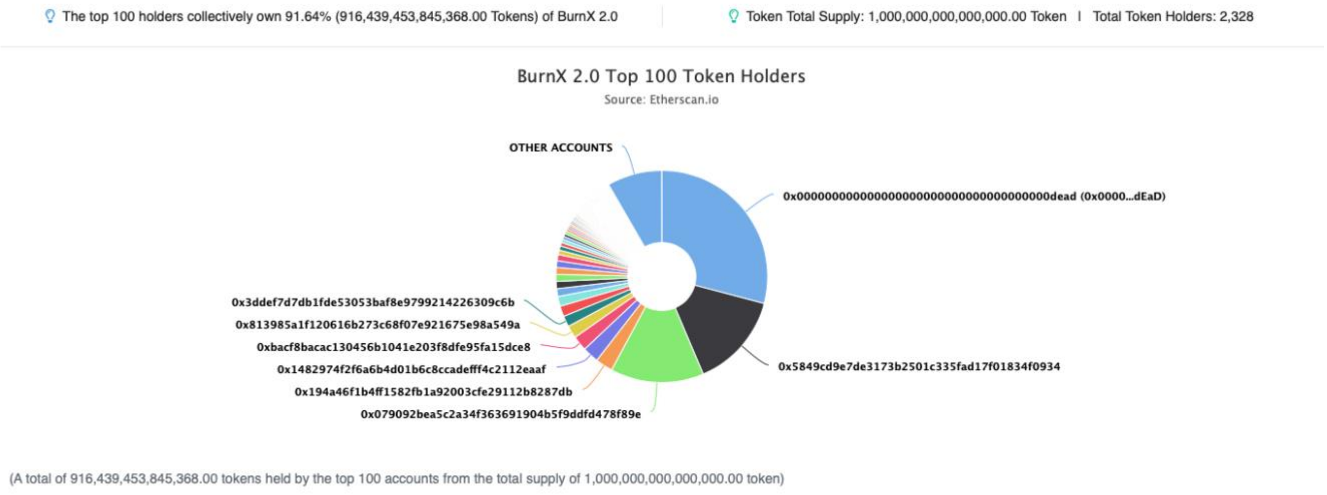
The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 19.07.2021

Contract name	BURNX 2.0
Contract address	0x1e950AF2F6f8505c09F0Ca42c4b38F10979cb22E
Total supply	1,000,000,000,000,000
Token ticker	BurnX20
Decimals	9
Token holders	2,328
Transactions count	5,646
Top 100 holders dominance	91.64%
Default Liquidity fee	4
Default Tax fee	4
Total fees	3812127422909160077999
Uniswap V2 pair	0xaf7c6dead245b93de19bb1bb828b0acce94aefb3
Contract deployer address	0x40Fa8dF858C801Cf0258121d9977d4292810267A
Contract's current owner address	0x40fa8df858c801cf0258121d9977d4292810267a

BURNX 2.0 Token Distribution



The top 100 holders collectively own 91.64% (916,439,453,845,368.00 Tokens) of BurnX 2.0

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 2,328

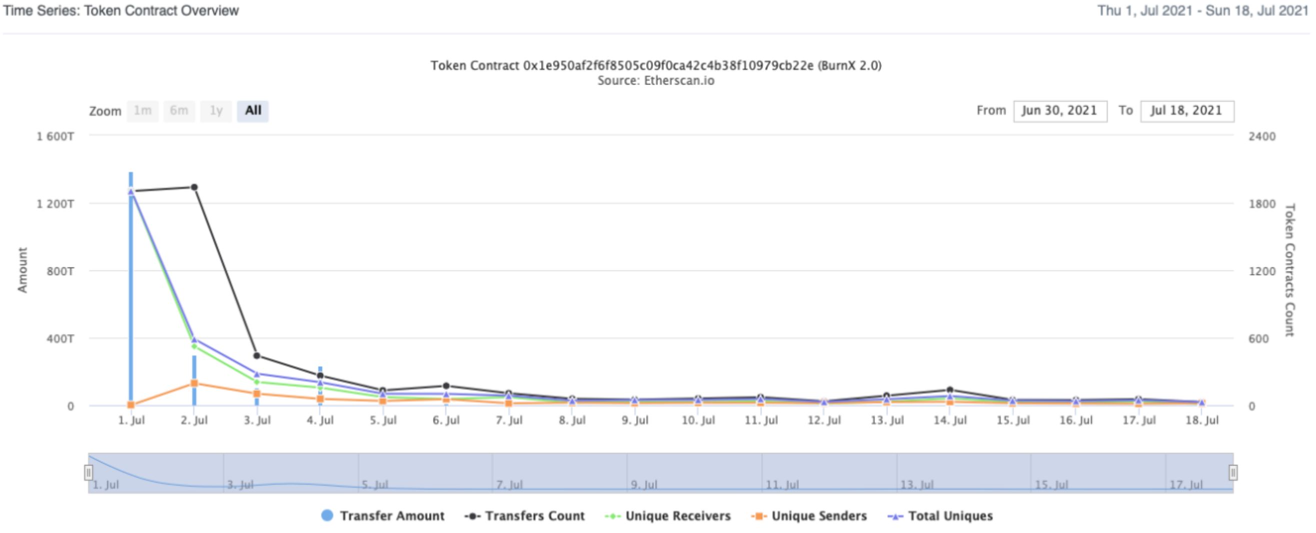
BurnX 2.0 Top 100 Token Holders

Source: Etherscan.io

OTHER ACCOUNTS
0x000000000000000000000000000000000000dead (0x0000...dEaD)
0x3ddef7d7db1fde53053baf8e9799214226309c6b
0x813985a1f120616b273c68f07e921675e98a549a
0xbacf8bacac130456b1041e203f8dfe95fa15dce8
0x1482974f2f6a6b4d01b6c8ccadefff4c2112eaaf
0x194a46f1b4ff1582fb1a92003cfe29112b8287db
0x079092bea5c2a34f363691904b5f9ddfd478f89e
0x5849cd9e7de3173b2501c335fad17f01834f0934

(A total of 916,439,453,845,368.00 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000,000.00 token)

BURNX 2.0 Contract Interaction Details



Time Series: Token Contract Overview

Thu 1, Jul 2021 - Sun 18, Jul 2021

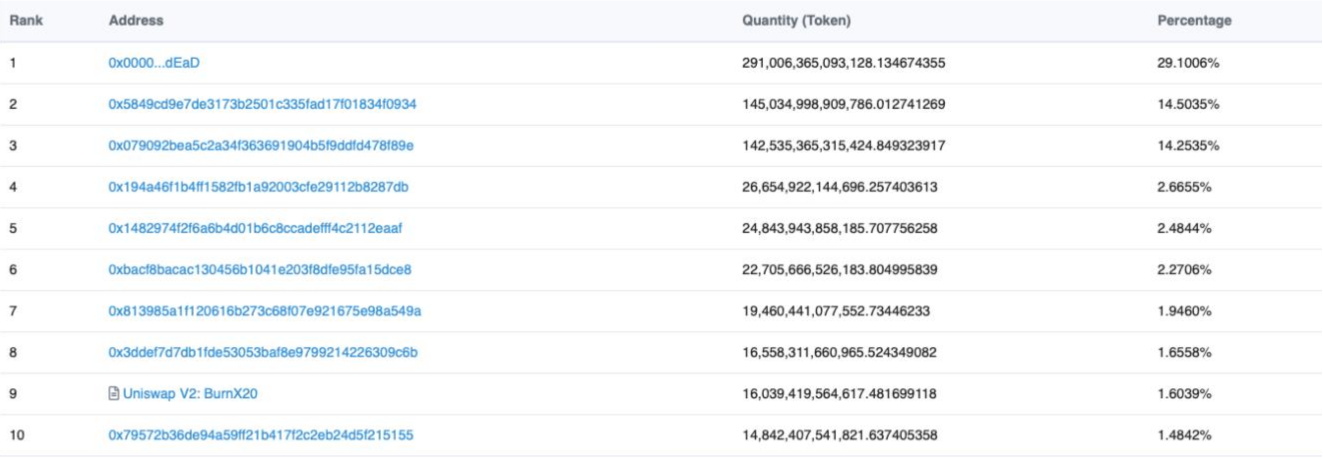
Token Contract 0x1e950af2f6f8505c09f0ca42c4b38f10979cb22e (BurnX 2.0)

Source: Etherscan.io

Zoom 1m 6m 1y All

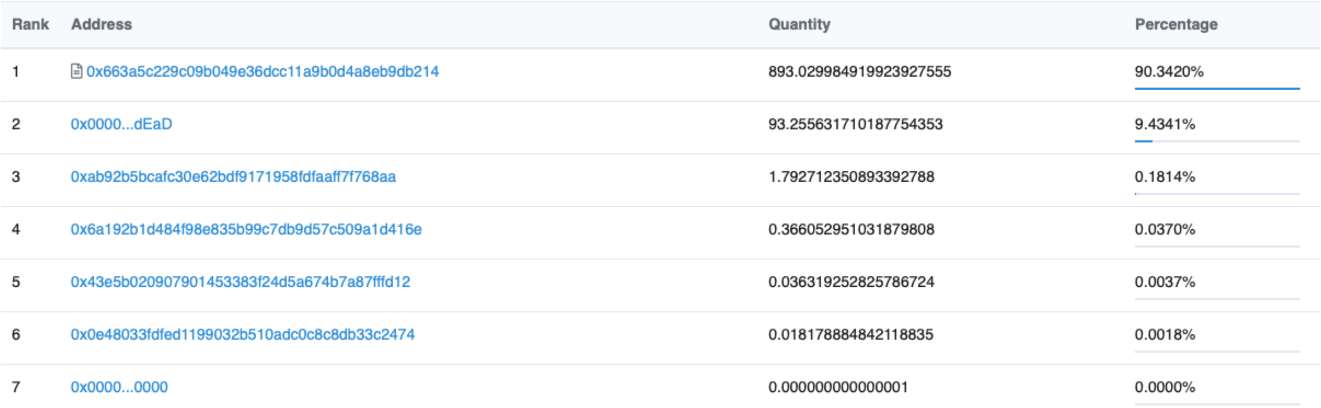
From Jun 30, 2021 To Jul 18, 2021

BURNX 2.0 Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	0x0000...dEaD	291,006,365,093,128.134674355	29.1006%
2	0x5849cd9e7de3173b2501c335fad17f01834f0934	145,034,998,909,786.012741269	14.5035%
3	0x079092bea5c2a34f363691904b5f9ddfd478f89e	142,535,365,315,424.849323917	14.2535%
4	0x194a46f1b4ff1582fb1a92003cfe29112b8287db	26,654,922,144,696.257403613	2.6655%
5	0x1482974f2f6a6b4d01b6c8ccadefff4c2112eaaf	24,843,943,858,185.707756258	2.4844%
6	0xbacf8bacac130456b1041e203f8dfe95fa15dce8	22,705,666,526,183.804995839	2.2706%
7	0x813985a1f120616b273c68f07e921675e98a549a	19,460,441,077,552.73446233	1.9460%
8	0x3ddef7d7db1fde53053baf8e9799214226309c6b	16,558,311,660,965.524349082	1.6558%
9	Uniswap V2: BurnX20	16,039,419,564,617.481699118	1.6039%
10	0x79572b36de94a59ff21b417f2c2eb24d5f215155	14,842,407,541,821.637405358	1.4842%

BURNX 2.0 LP Token Holders



Rank	Address	Quantity	Percentage
1	0x663a5c229c09b049e36dcc11a9b0d4a8eb9db214	893.029984919923927555	90.3420%
2	0x0000...dEaD	93.255631710187754353	9.4341%
3	0xab92b5bcafc30e62bdf9171958fdfaaff7f768aa	1.792712350893392788	0.1814%
4	0x6a192b1d484f98e835b99c7db9d57c509a1d416e	0.366052951031879808	0.0370%
5	0x43e5b020907901453383f24d5a674b7a87fffd12	0.036319252825786724	0.0037%
6	0x0e48033fdfed1199032b510adc0c8c8db33c2474	0.018178884842118835	0.0018%
7	0x0000...0000	0.000000000000001	0.0000%

Contract functions details

```
+ [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

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

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

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

+ BurnX20 (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
  - [Ext] setExcludeFromFee #
    - modifiers: onlyOwner
  - [Pub] totalFees
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Ext] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Ext] addBots #
    - modifiers: onlyOwner
  - [Ext] removeBot #
    - modifiers: onlyOwner
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Pub] isExcludedFromFee
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] swapAndLiquify #
    - modifiers: noReentrant
  - [Prv] swapTokensForEth #
  - [Prv] addLiquidity #
  - [Prv] sendETHToMarketing #
  - [Ext] manualSwap #
    - modifiers: onlyOwner
  - [Pub] manualSend #
    - modifiers: onlyOwner
  - [Ext] setSwapLiquifyEnabled #
    - modifiers: onlyOwner
  - [Pub] isSwapLiquifyEnabled
  - [Prv] _tokenTransfer #
  - [Prv] _transferStandard #
  - [Prv] _transferToExcluded #
  - [Prv] _transferFromExcluded #
  - [Prv] _transferBothExcluded #
  - [Prv] _takeMarketingLiquidity #
  - [Prv] _reflectFee #

- [Ext] <Fallback> ($)
- [Prv] _getValues
- [Prv] _getTValues
- [Prv] _getRValues
- [Prv] _getRate
- [Prv] _getCurrentSupply
- [Ext] setTxFees #
  - modifiers: onlyOwner
- [Ext] setWallets #
  - modifiers: onlyOwner
- [Ext] setAmountSellLiquidity #
  - modifiers: onlyOwner
- [Ext] setMaxTx #
  - modifiers: onlyOwner
- [Pub] recoverTokens #
  - modifiers: onlyOwner
- [Ext] withdrawToken #
  - modifiers: onlyOwner
- [Ext] migrateHolders #
  - modifiers: onlyOwner

($) = 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.	Medium issue
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

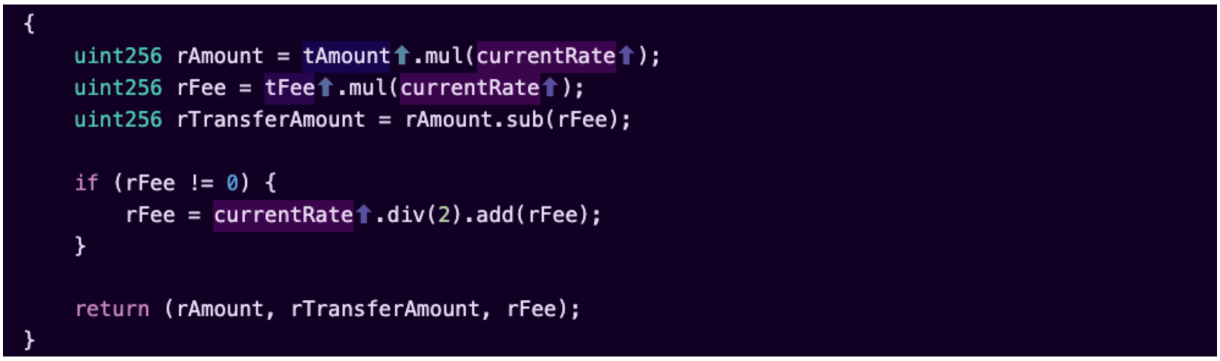
No high severity issues found.

Medium Severity Issues

1. Dynamic rFee calculation

Issue:

- The function _getRValues() changes rFee after subtracting it from rAmount. After that, there will be less correlation between rFee and tFee. Every second transaction will oversize total token balance by 1, so after some time, sum of the users' balances won't equal to total supply.



```
{
    uint256 rAmount = tAmount.mul(currentRate);
    uint256 rFee = tFee.mul(currentRate);
    uint256 rTransferAmount = rAmount.sub(rFee);

    if (rFee != 0) {
        rFee = currentRate.div(2).add(rFee);
    }

    return (rAmount, rTransferAmount, rFee);
}
```

Recommendation:

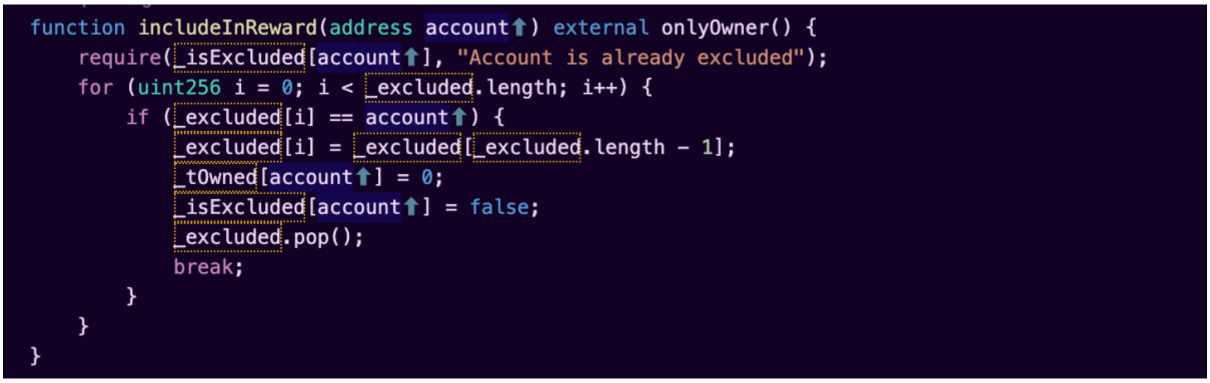
Check that changing rFee is really needed or change it before transfer amount calculations.

Low Severity Issues

2. Out of gas

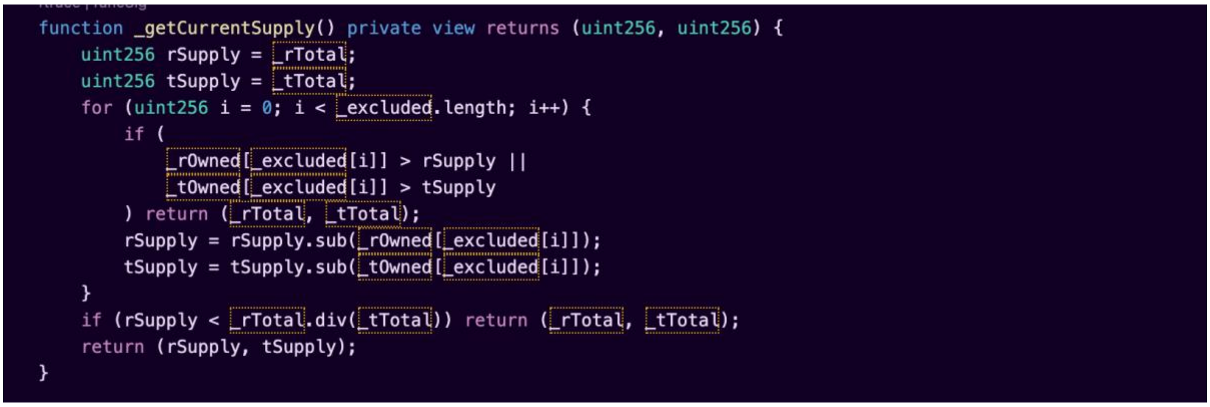
Issue:

- 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.



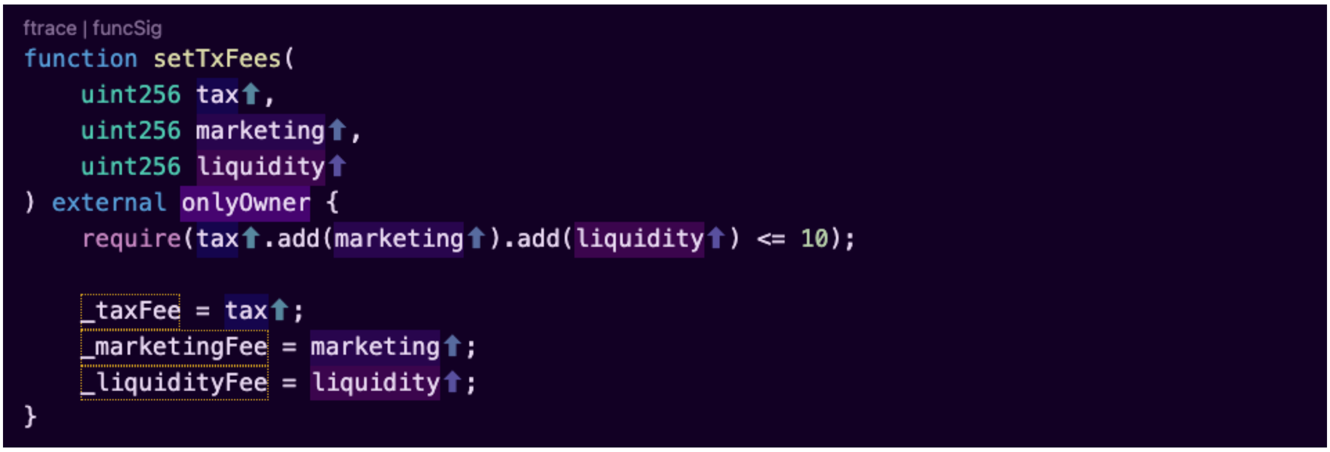
```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
      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

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

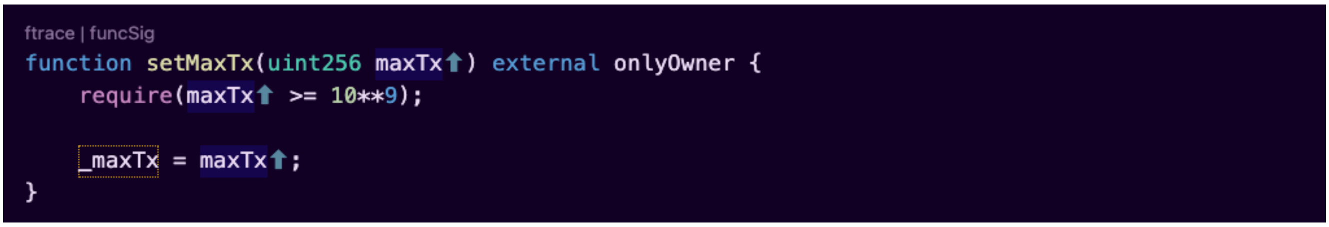
- Owner can change fees.



ftrace | funcSig

```
function setTxFees(
    uint256 tax,
    uint256 marketing,
    uint256 liquidity
) external onlyOwner {
    require(tax.add(marketing).add(liquidity) <= 10);

    _taxFee = tax;
    _marketingFee = marketing;
    _liquidityFee = liquidity;
}
```

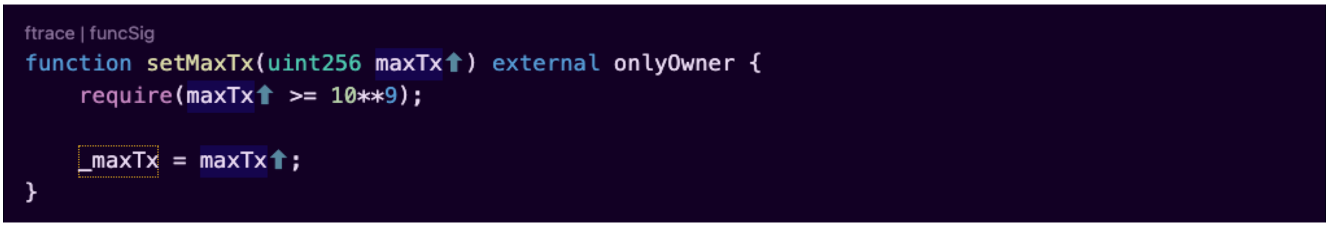
- Owner can change the maximum transaction amount.



ftrace | funcSig

```
function setMaxTx(uint256 maxTx) external onlyOwner {
    require(maxTx >= 10**9);

    _maxTx = maxTx;
}
```

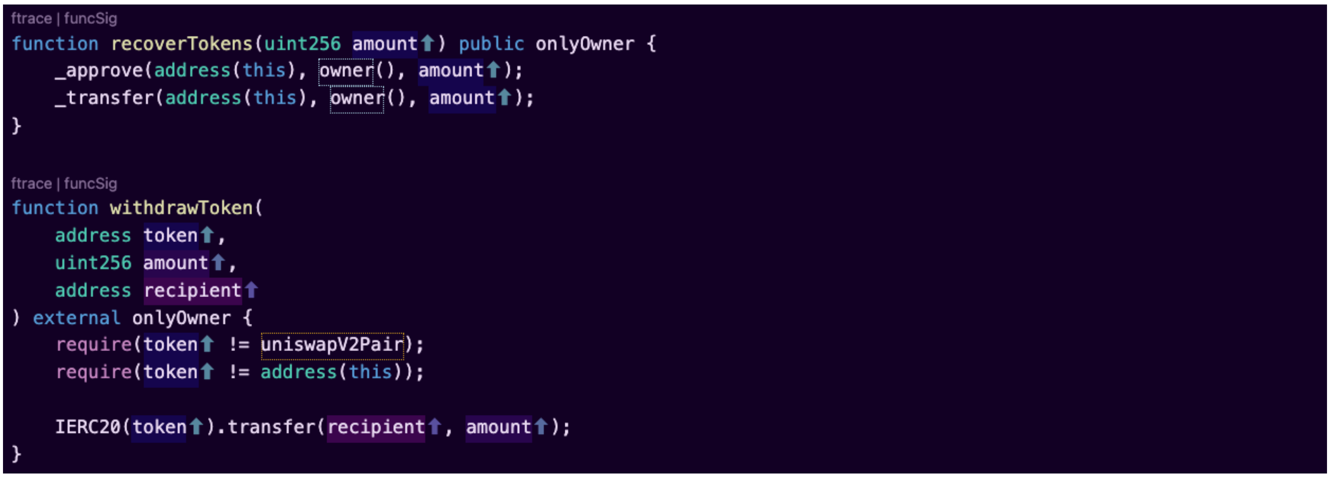
- Owner can call migrate holders function.



ftrace | funcSig

```
function setMaxTx(uint256 maxTx) external onlyOwner {
    require(maxTx >= 10**9);

    _maxTx = maxTx;
}
```

- Owner can withdraw tokens.



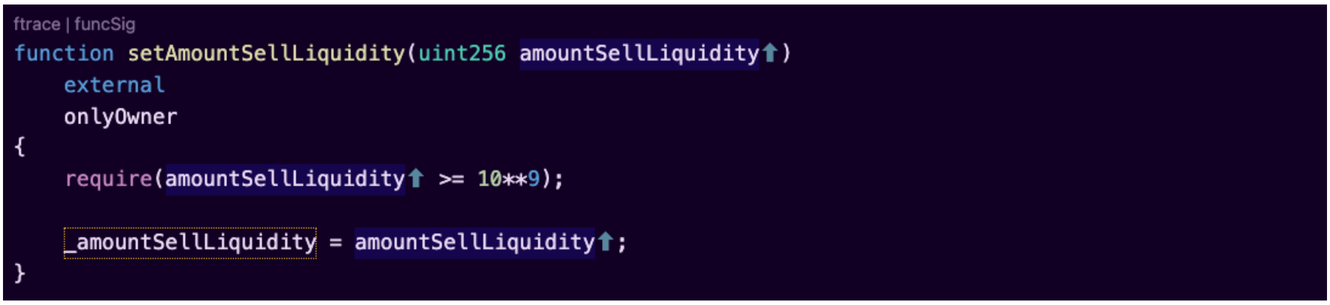
ftrace | funcSig

```
function recoverTokens(uint256 amount) public onlyOwner {
    _approve(address(this), owner(), amount);
    _transfer(address(this), owner(), amount);
}
```

ftrace | funcSig

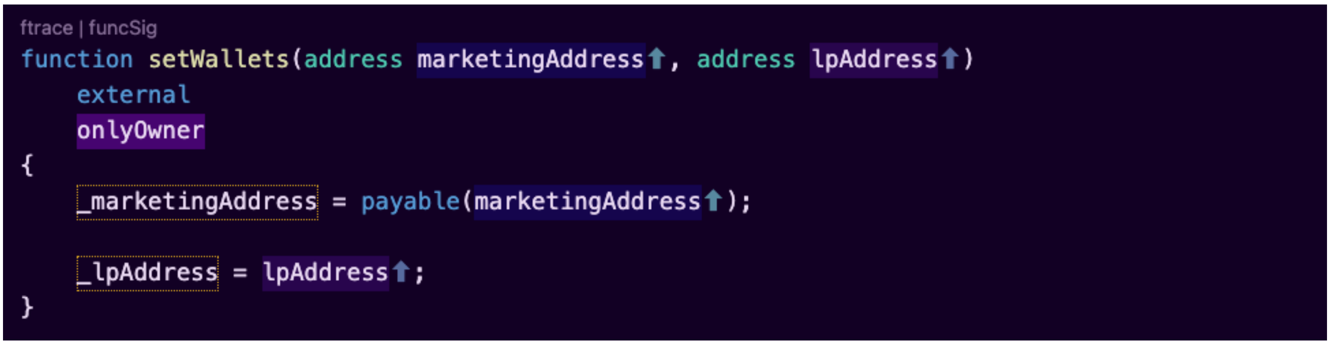
```
function withdrawToken(
    address token,
    uint256 amount,
    address recipient
) external onlyOwner {
    require(token != uniswapV2Pair);
    require(token != address(this));

    IERC20(token).transfer(recipient, amount);
}
```

- Owner can change amount sell liquidity.



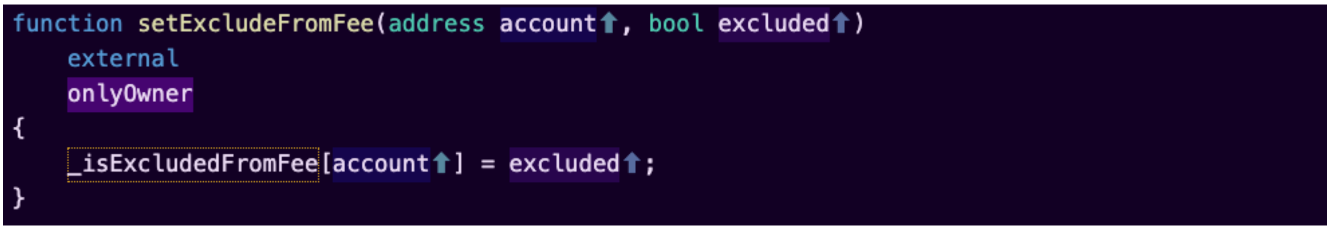
```
function setAmountSellLiquidity(uint256 amountSellLiquidity)
    external
    onlyOwner
{
    require(amountSellLiquidity >= 10**9);

    _amountSellLiquidity = amountSellLiquidity;
}
```

- Owner can change wallets.



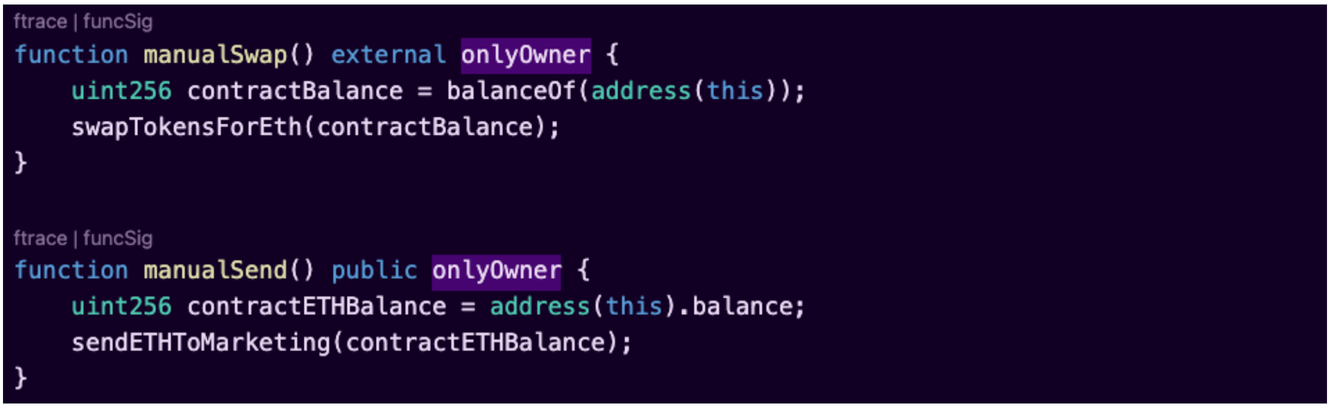
```
function setWallets(address marketingAddress, address lpAddress)
    external
    onlyOwner
{
    _marketingAddress = payable(marketingAddress);

    _lpAddress = lpAddress;
}
```

- Owner can exclude from the fee.



```
function setExcludeFromFee(address account, bool excluded)
    external
    onlyOwner
{
    _isExcludedFromFee[account] = excluded;
}
```

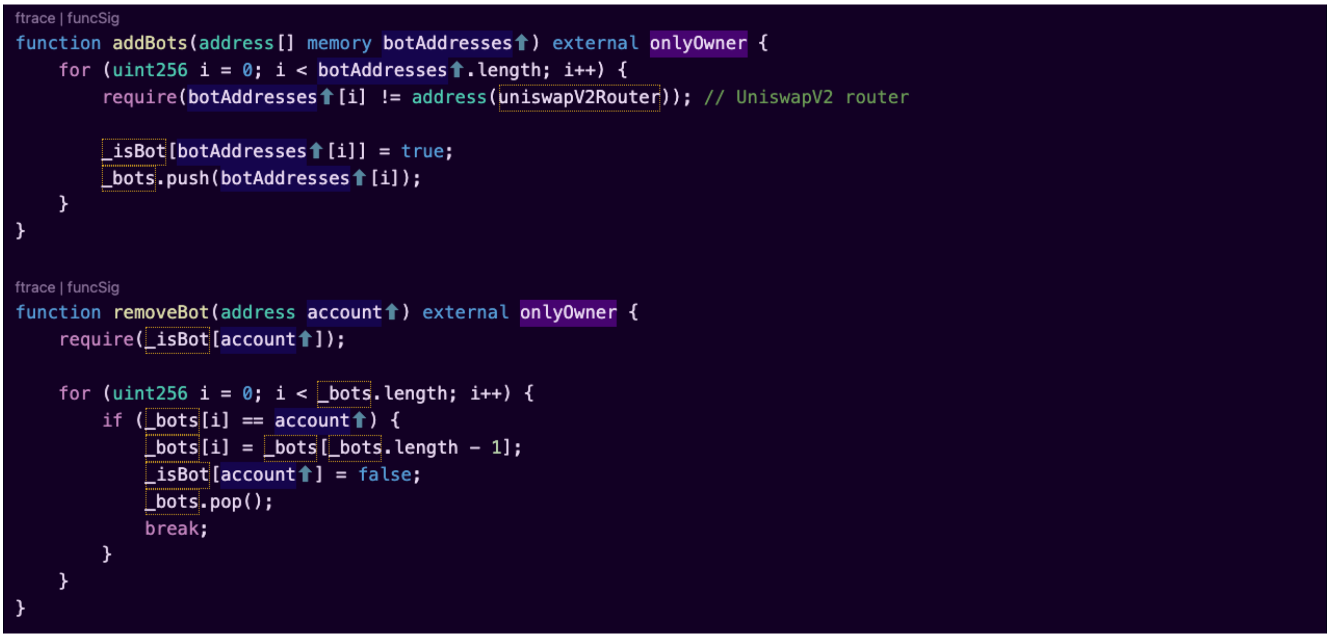
- Owner can manually swap contract balance and send it to marketing.



```
function manualSwap() external onlyOwner {
    uint256 contractBalance = balanceOf(address(this));
    swapTokensForEth(contractBalance);
}
```

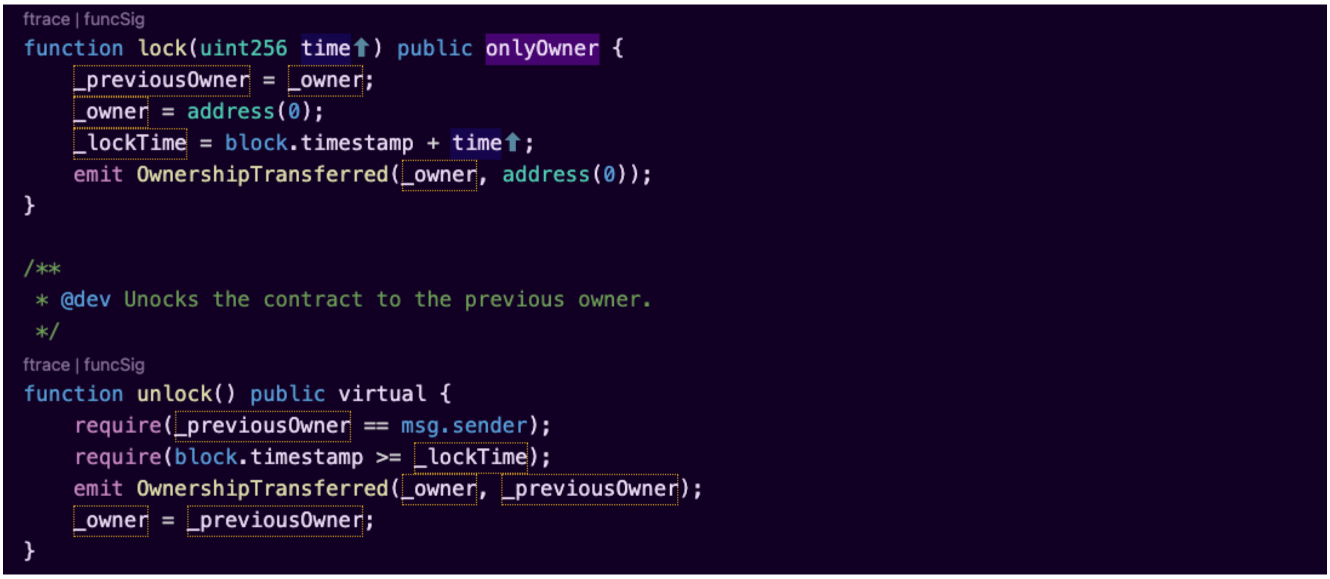
```
function manualSend() public onlyOwner {
    uint256 contractETHBalance = address(this).balance;
    sendETHToMarketing(contractETHBalance);
}
```

- Owner can add and remove bots.



```
ftrace | funcSig
function addBots(address[] memory botAddresses) external onlyOwner {
    for (uint256 i = 0; i < botAddresses.length; i++) {
        require(botAddresses[i] != address(uniswapV2Router)); // UniswapV2 router

        _isBot[botAddresses[i]] = true;
        _bots.push(botAddresses[i]);
    }
}

ftrace | funcSig
function removeBot(address account) external onlyOwner {
    require(_isBot[account]);

    for (uint256 i = 0; i < _bots.length; i++) {
        if (_bots[i] == account) {
            _bots[i] = _bots[_bots.length - 1];
            _isBot[account] = false;
            _bots.pop();
            break;
        }
    }
}
```

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



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

/**
 * @dev Unocks the contract to the previous owner.
 */
ftrace | funcSig
function unlock() public virtual {
    require(_previousOwner == msg.sender);
    require(block.timestamp >= _lockTime);
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

Conclusion

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

Liquidity locking details provided by the team:
https://app.unicrypt.network/amm/uni-v2/pair/0xaF7C6DeAd245b93dE19BB1BB828B0AcCE94AEfb3

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.