CheckPoint

Token Security Audit Report
Prepared for LittleFlokiFrunk

[v.1.0]

October 2021

Document Properties

Client	LittleFlokiFrunk
Platform	Binance Smart Chain
Language	Solidity
Codebase	0xE8c0D746E587e36aCa81eA979543F31921BfddFC

Audit Summary

Delivery Date	11.10.2021
Audit Methodology	Static Analysis, Manual Review
Auditor(s)	Erno Patiala
Classification	Publlic
Version	1.0

Contact Information

Company	CheckPoint
Name	Hanna Järvinen
Telegram	t.me/checkpointreport
E-mail	contact@checkpoint.report

Remark: For more information about this document and its contents, please contact CheckPoint team

Table Of Contents

1 Executive Summary

On 11/10/2021, CheckPoint conducted a full audit for the LittleFlokiFrunk to verify the overall security posture including a smart contract review to discover issues and vulnerabilities in the source code. Static Code Analysis, Dynamic Analysis, and Manual Review werdone in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem.

After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to. More information can be found in **Section 5 'Audit Result'**. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

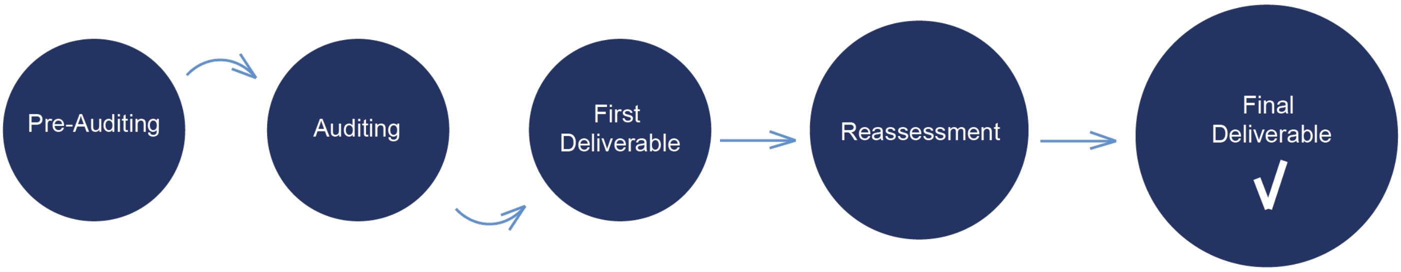
🟩 **LittleFlokiFrunk (DYNA)**

Low Risk Level

Communication Channels	Website Content Analysis, Social Media Listening
Smart Contract Code	Smart Contract Details, Contract Function Details, Issues Checking Status, Detailed Findings Information

THIS TOKEN PASSES CHECKPOINT'S SECURITY VERIFICATION STANDART

2 Audit Methodology



CheckPoint conducts the following procedure to enhance the security level of our clients' tokens:

- **Pre-Auditing**

  Planning a comprehensive survey of the token, its ecosystem, possible risks & prospects, getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing.

- **Auditing**

  Study of all available information about the token on the Web, inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals.

- **First Deliverable and Consulting**

  Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation.

- **Reassessment**

  Verifying the status of the issues and whether there are any other complications in the fixes applied.

- **Final Deliverable**

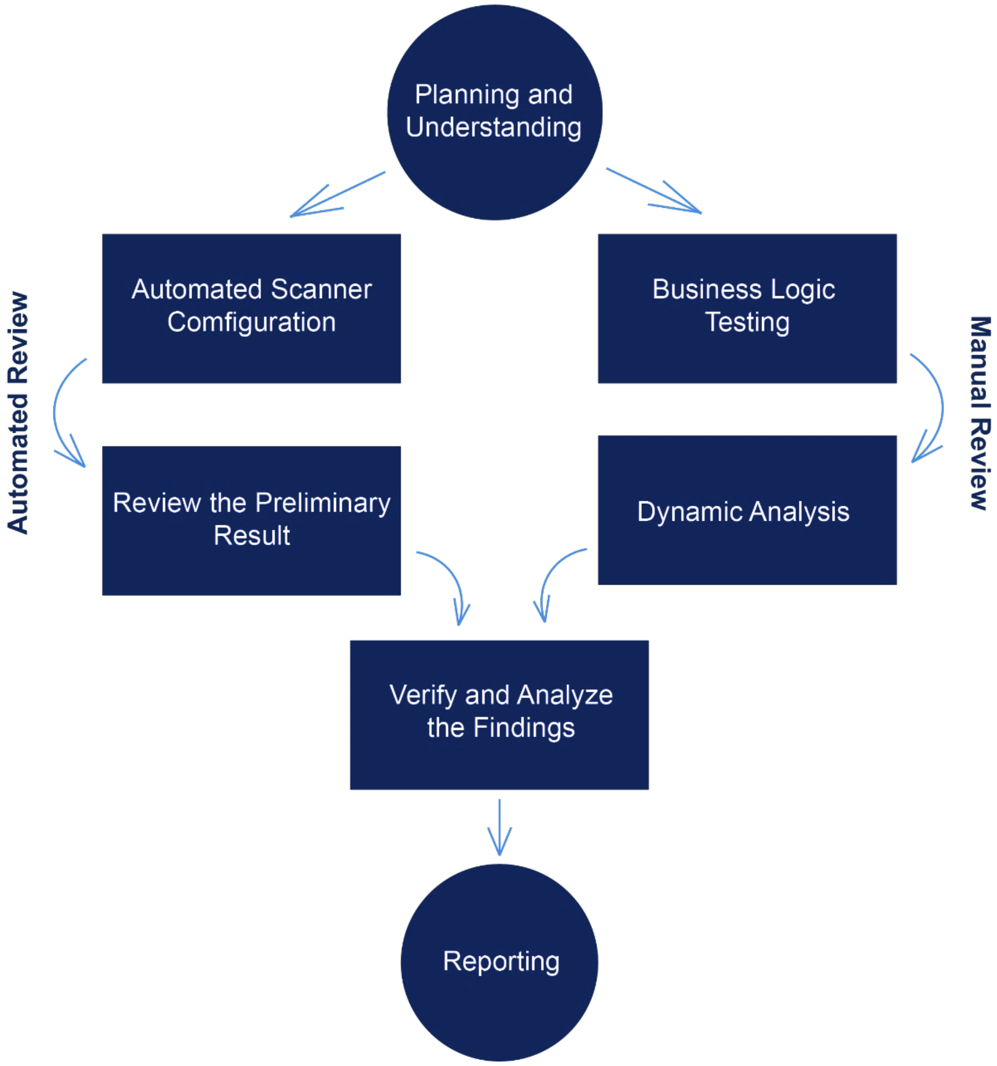
  Providing a full report with the detailed status of each issue.

The security audit process of CheckPoint includes three types testing:

1. Examining publicly available information about the token on social networks, including a detailed overview of the official website and analysis of the latest messages and opinions about the token.

2. Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

3. Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.


Planning and Understanding
Automated Scanner Comfiguration
Business Logic Testing
Automated Review
Manual Review
Review the Preliminary Result
Dynamic Analysis
Verify and Analyze the Findings
Reporting

Remark: Manual and Automated review approaches can be mixed and matched including business logic analysis in terms of malicious doers' perspective

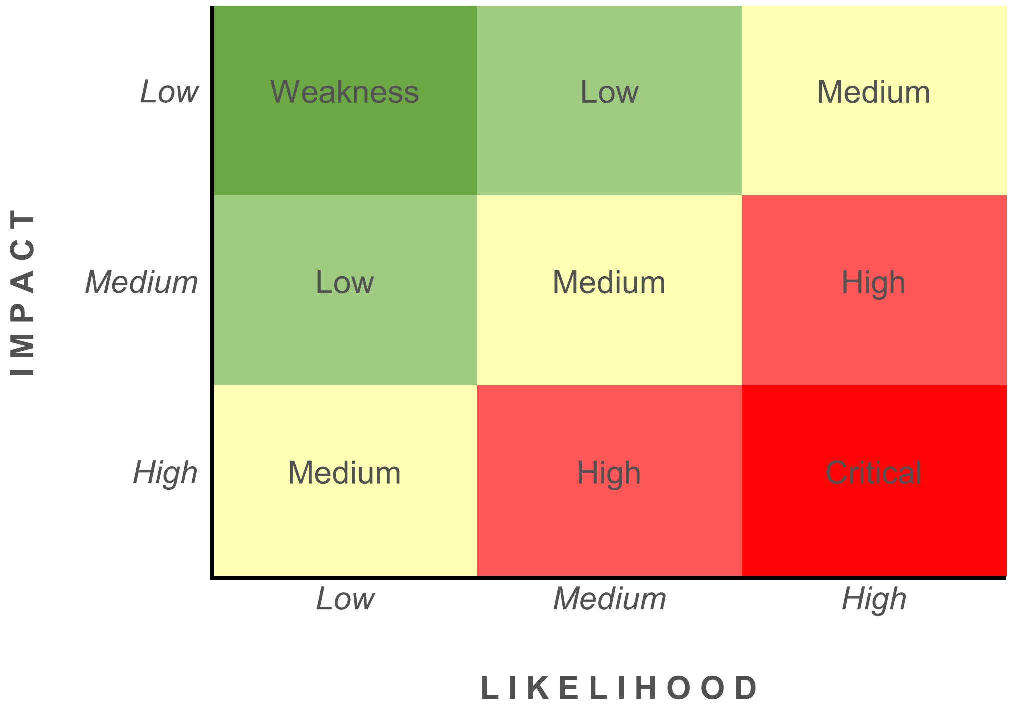
In particular, we perform the audit according to the following procedure:

- **Planning & Understanding**
  - determine scope of testing and understand application purpose and workflows;
  - identify key risk areas, including technical and business risks;
  - determine approach – which sections to review within the resource constraints and review method – automated, manual or mixed.
- **Automated Review**
  - adjust automated source code review tools to inspect the code for known unsafe coding patterns;
  - verify output of the tool in order to eliminate false positive result, and if necessary, adjust and re-run the code review tool.
- **Manual Review**
  - testing for business logic flaws requires thinking in unconventional methods;
  - identify unsafe coding behavior via static code analysis.
- **Reporting**
  - analyze the root cause of the flaws;
  - recommend coding process improvements.

3 Risk Level Classification

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- **Likelihood** represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- **Impact** measures the technical loss and business damage of a successful attack.
- **Severity** demonstrates the overall criticality of the risk and calculated as the product of impact and likelihood values, illustrated in a twodimensional matrix. The shading of the matrix visualizes the different risk levels.



IMPACT	Low	Medium	High
Low	Weakness	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

LIKELIHOOD

Remark: Likelihood and Impact are categorized into three levels: H, M, and L, i.e., High, Medium and Low respectively. Severity is determined by likelihood and impact and can be classified into five categories accordingly, i.e., Critical, High, Medium, Low and Weakness

For prioritization of the vulnerabilities, we have adopted the scheme by five distinct levels for risk: Critical, High, Medium, Low, and Weakness. The risk level definitions are presented in table.

LEVEL	DESCRIPTION
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project

4 Project Overview

4.1 Communication Channels

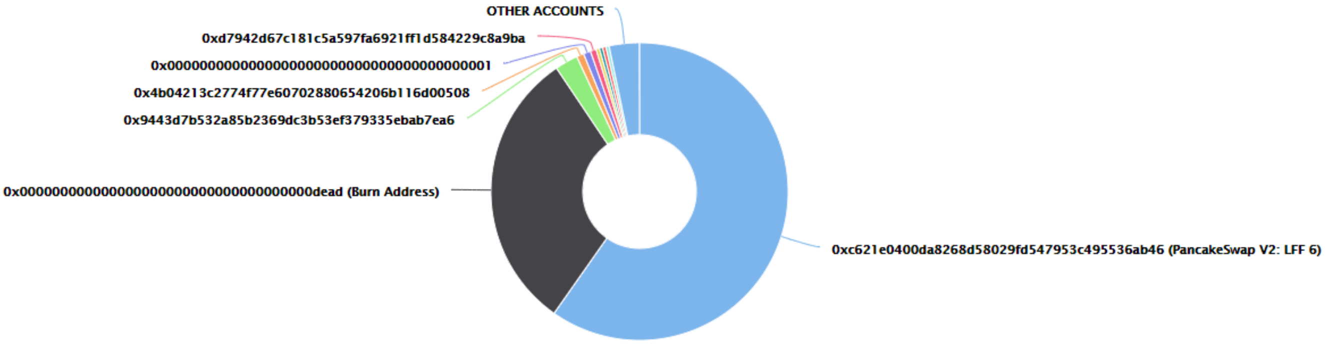
- ✓ No Website [RISK]
- ✓ 2 Social Media Networks
- ✓ 1000+ Telegram Members
- ✓ 2000+ Twitter Followers
- ✓ No Injected Spam and Popus Found
- ✓ No Active Voice Chats [RISK]

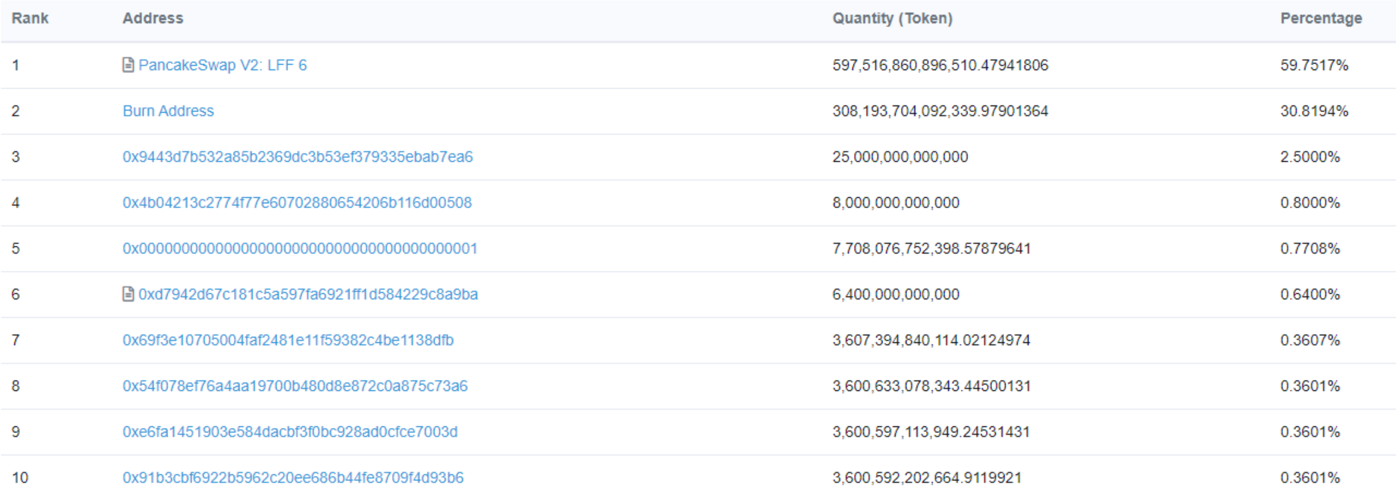
Remark: This page contains active links

4.2 Smart Contract Details

Contract Name	LittleFlokiFrunk
Contract Address	0xE8c0D746E587e36aCa81eA979543F31921BfddFC
Total Supply	1,000,000,000,000,000
Token Ticker	LFF
Decimals	8
Token Holders	116
Transactions Count	857
Top 10 Holders Dominance	96,72%
Charity Fee	0%
Liquidity Fee	4%
Tax Fee	4%
Marketing And Dev Fee	2%
Uniswap V2 Pair Contract	0xc621e0400da8268d58029fd547953c495536ab46
Contract Deployer Address	0x82895Ae42E5e62955DB2F3546f9E21BD6a0f4F5B
Current Owner Address	0x82895Ae42E5e62955DB2F3546f9E21BD6a0f4F5B

LittleFlokiFrunk Top 10 Token Holders



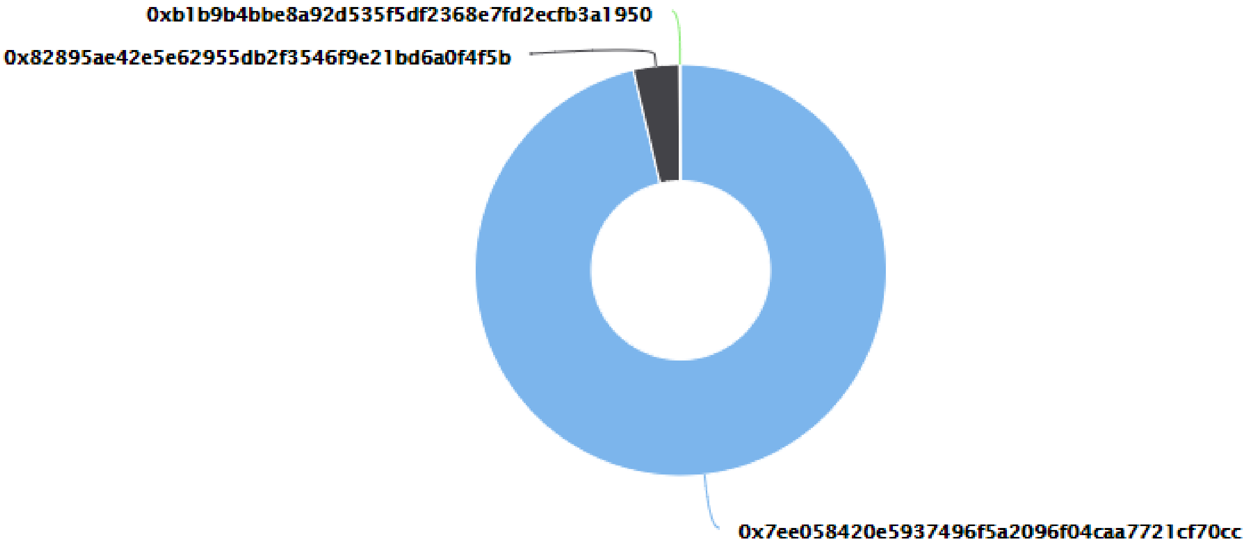


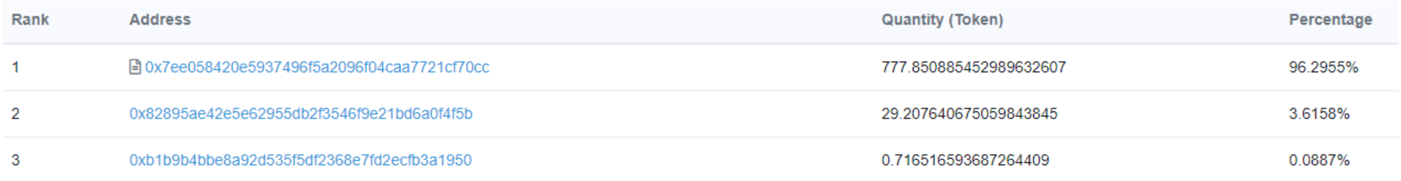
Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: LFF 6	597,516,860,896,510.47941806	59.7517%
2	Burn Address	308,193,704,092,339.97901364	30.8194%
3	0x9443d7b532a85b2369dc3b53ef379335ebab7ea6	25,000,000,000,000	2.5000%
4	0x4b04213c2774f77e60702880654206b116d00508	8,000,000,000,000	0.8000%
5	0x0000000000000000000000000000000000000001	7,708,076,752,398.57879641	0.7708%
6	0xd7942d67c181c5a597fa6921ff1d584229c8a9ba	6,400,000,000,000	0.6400%
7	0x69f3e10705004faf2481e11f59382c4be1138dfb	3,607,394,840,114.02124974	0.3607%
8	0x54f078ef76a4aa19700b480d8e872c0a875c73a6	3,600,633,078,343.44500131	0.3601%
9	0xe6fa1451903e584dacbf3f0bc928ad0cfce7003d	3,600,597,113,949.24531431	0.3601%
10	0x91b3cbf6922b5962c20ee686b44fe8709f4d93b6	3,600,592,202,664.9119921	0.3601%

✓ ~31% tokens are permanently removed from circulation

✓ PancakeSwap holds ~60% of the token's supply as liquidity

LittleFlokiFrunk Top 3 LP Token Holders

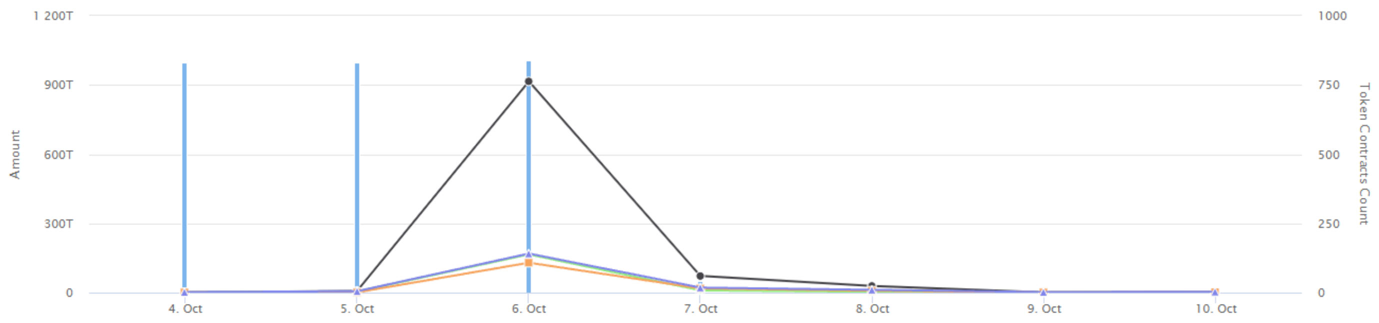




Rank	Address	Quantity (Token)	Percentage
1	0x7ee058420e5937496f5a2096f04caa7721cf70cc	777.850885452989632607	96.2955%
2	0x82895ae42e5e62955db2f3546f9e21bd6a0f4f5b	29.207640675059843845	3.6158%
3	0xb1b9b4bbe8a92d535f5df2368e7fd2ecfb3a1950	0.716516593687264409	0.0887%

[RISK] 1 wallet have ~96,0% LP tokens

LittleFlokiFrunk Contract Interaction Details



4.3 Contract Function Details

```
$ = payable function
# = non-constant function
[Int] = Internal
[Pub] = Public
[Prv] = Private
[Ext] = External

+ [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] firstOwner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner
   - [Pub] geUnlockTime
```

```
    - [Pub] lock #
       - modifiers: onlyOwner

 + [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 #

 + LittleFlokiFrunk (Context, IERC20, Ownable)
   - [Pub] <Constructor> #
   - [Pub] lockTimeOfWallet
   - [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] deliver #
   - [Pub] reflectionFromToken
   - [Pub] tokenFromReflection
   - [Pub] excludeFromReward #
      - modifiers: onlyOwner
   - [Ext] includeInReward #
      - modifiers: onlyOwner
   - [Prv] _transferBothExcluded #
   - [Pub] excludeFromFee #
      - modifiers: onlyOwner
   - [Pub] setCharityAddress #
      - modifiers: onlyOwner

   - [Pub] setMarketingDevAddress #
```

```
    - modifiers: onlyOwner
 - [Pub] showMarketingDevAddress $
 - [Pub] showCharityAddress $
 - [Pub] includeInFee
    - modifiers: onlyOwner
 - [Ext] setCharityFeePercent #
    - modifiers: onlyOwner
 - [Ext] setTaxFeePercent #
    - modifiers: onlyOwner
 - [Ext] setMarketingDevFeePercent #
    - modifiers: onlyOwner
 - [Ext] seLiquidityFeePercent #
    - modifiers: onlyOwner
 - [Ext] setMaxTxPercent #
    - modifiers: onlyOwner
 - [Ext] setSwapAndLiquifyEnabled #
    - modifiers: onlyOwner
 - [Ext] preparePresale #
    - modifiers: onlyOwner
 - [Ext] afterPresale #
    - modifiers: onlyOwner
 - [Prv] _reflectFee #
 - [Prv] _getValues
 - [Prv] _getTValues
 - [Prv] _getRValues
 - [Prv] _getRate
 - [Prv] _getCurrentSupply
 - [Prv] _takeLiquidity #
 - [Prv] calculateTaxFee
 - [Prv] calculateLiquidityFee
 - [Prv] removeAllFee #
 - [Prv] restoreAllFee #
 - [Pub] isExcludedFromFee
 - [Prv] _approve #
 - [Prv] _transfer #
 - [Prv] swapAndLiquify #
    - modifiers: lockTheSwap
 - [Prv] swapTokensForEth #
 - [Prv] addLiquidity #
 - [Prv] _tokenTransfer #
 - [Prv] _transferStandard #
 - [Prv] _transferToExcluded #
 - [Prv] _transferFromExcluded #
```

4.4 Issues Checking Status

CHECKING ITEM	NOTES	RESULT
Arbitrary Jump with Function Type Variable	N / A	PASS
Arithmetic Accuracy Deviation	N / A	PASS
Assert Violation	N / A	PASS
Authorization through tx.origin	N / A	PASS
Business Logic	N / A	PASS
Code with No Effects	N / A	PASS
Critical Solidity Compiler	N / A	PASS
Delegatecall to Untrusted Callee	N / A	PASS
Design Logic	N / A	PASS
DoS with Block Gas Limit	N / A	PASS
DoS with Failed Call	N / A	PASS
Function Default Visibility	N / A	PASS
Hash Collisions With MVLA	N / A	PASS
Incorrect Constructor Name	N / A	PASS
Incorrect Inheritance Order	N / A	PASS
Integer Overflows and Underflows	N / A	PASS
Lack of Proper Signature Verification	N / A	PASS
Message Call with Hardcoded Gas Amount	N / A	PASS
Missing Protection Against SRA	N / A	PASS
Presence of Unused Variables	N / A	PASS
Reentrancy	N / A	PASS
Requirement Violation	N / A	PASS

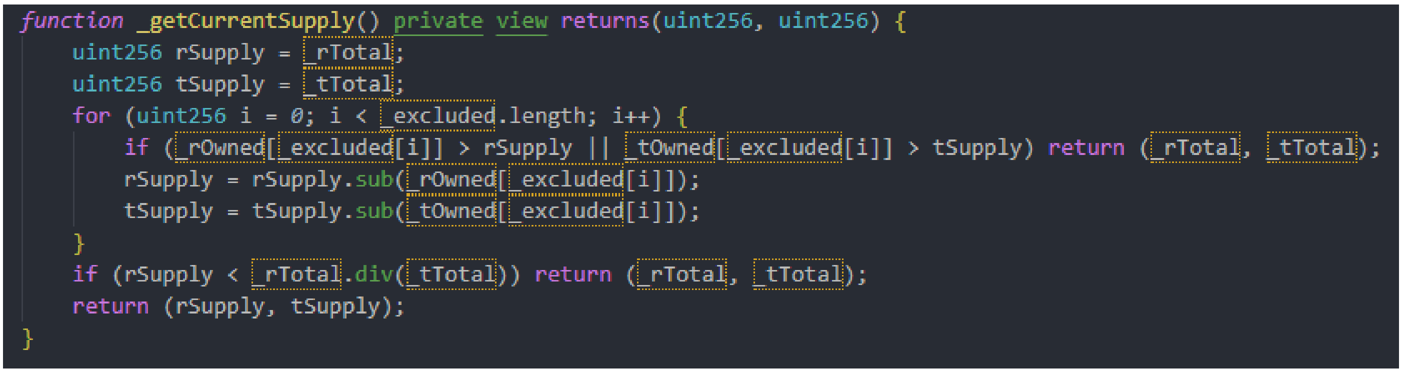
CHECKING ITEM	NOTES	RESULT
Right-To-Left-Override Control Character	N / A	PASS
Shadowing State Variables	N / A	PASS
Signature Malleability	N / A	PASS
State Variable Default Visibility	N / A	PASS
Timestamp Dependence	N / A	PASS
Transaction Order Dependence	N / A	PASS
Typographical Error	N / A	PASS
Unencrypted Private Data On-Chain	N / A	PASS
Unexpected Ether balance	N / A	PASS
Uninitialized Storage Pointer	N / A	PASS
Use of Deprecated Solidity Functions	N / A	PASS
Weak Sources of Randomness From CA	N / A	PASS
Write to Arbitrary Storage Location	N / A	PASS

Remark: To evaluate the risk, we go through a list of check items and each would be labeled with a severity category. For one check item, if our tool or analysis does not identify any issue, the contract is considered safe regarding the check item

4.5 Detailed Findings Information

[RISK] DoS with Block Gas Limit

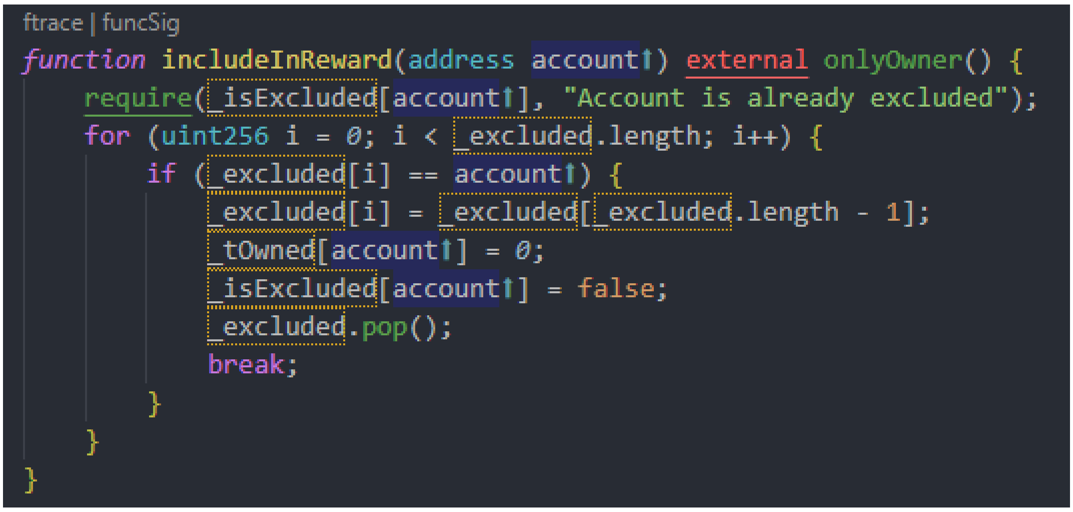
- The function _getCurrentSupply uses the loop for evaluating total supply and total reward. It could be aborted with out-of-gas exception if there will be a long excluded addresses list. Including an account in the reward again may result in unexpected behavior.



```
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: Consider removing the _getCurrentSupply function. If this is not desired, consider avoiding it, especially on accounts with a significant balance.

- The function includeInReward uses the loop to find and remove addresses from the _excluded list. It could be aborted with out-of-gas exception if there will be a long excluded addresses list. Including an account in the reward again may result in unexpected behavior.



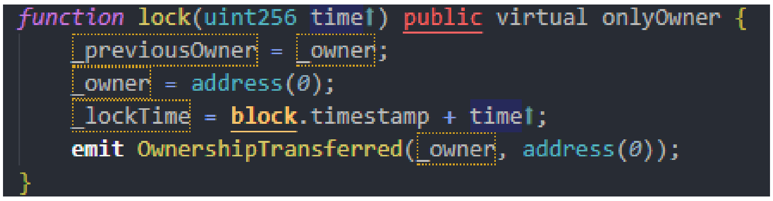
```
ftrace | funcSig
function includeInReward(address account↑) external onlyOwner() {
    require(_isExcluded[account↑], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

Recommendation: Consider removing the includeInReward function. If this is not desired, consider avoiding it, especially on accounts with a significant balance.

[RISK] Owner Privileges (in the period when the owner is not renounced)

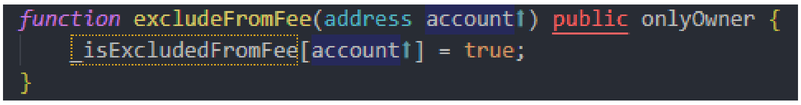
The contract contains the following privileged functions that are restricted by the onlyOwner.

- The owner of the contract can lock the contract.

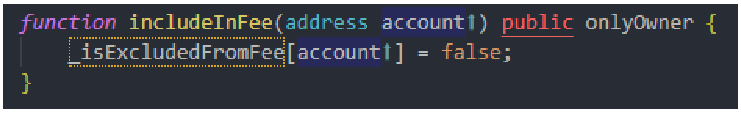


```
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}
```

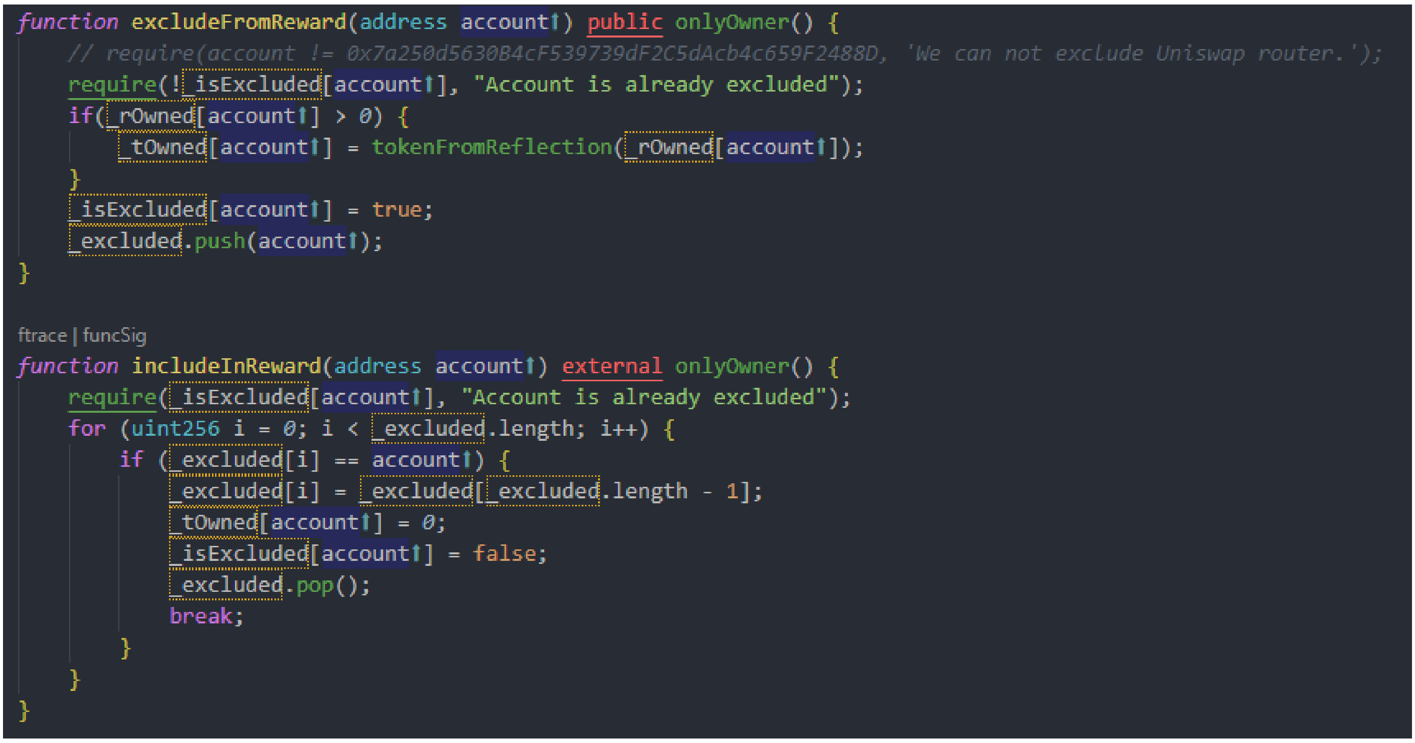
- The owner of the contract can exclude and include accounts from fees and reward distribution.



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

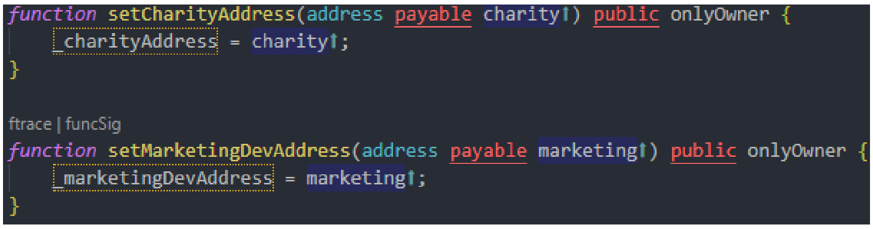


```
function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```



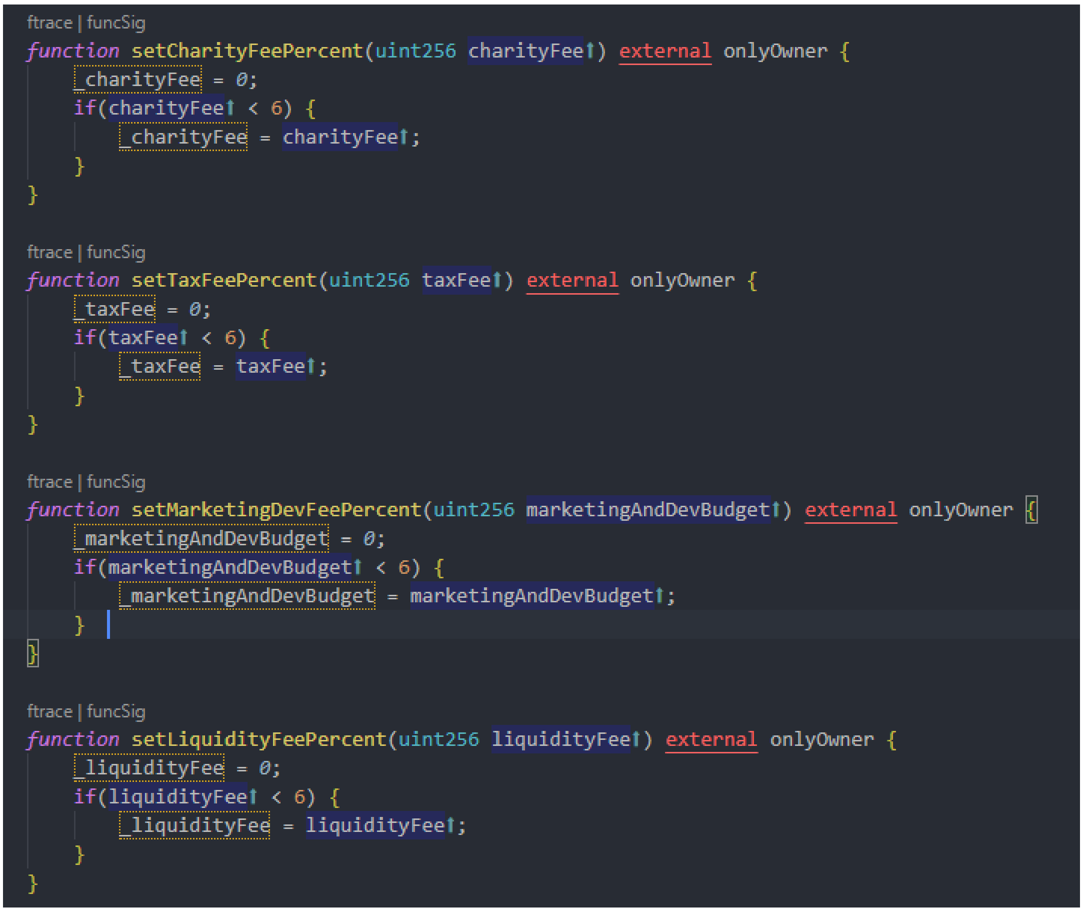
```
function excludeFromReward(address account) public onlyOwner() {
    // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap router.');
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

ftrace | funcSig
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 owner of the contract can change Charity and MarketingDev wallet addresses.



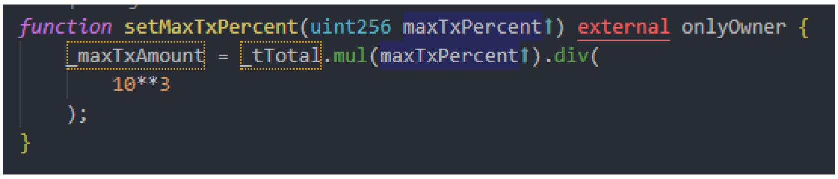
```
function setCharityAddress(address payable charity↑) public onlyOwner {
    _charityAddress = charity↑;
}

ftrace | funcSig
function setMarketingDevAddress(address payable marketing↑) public onlyOwner {
    _marketingDevAddress = marketing↑;
}
```

- The owner can set a ‘liquidity fee’, a ‘marketingdev fee’, a ‘tax fee’ and a ‘charity fee’.



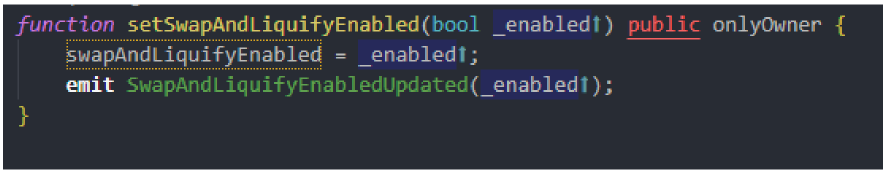
```
ftrace | funcSig
function setCharityFeePercent(uint256 charityFee↑) external onlyOwner {
    _charityFee = 0;
    if(charityFee↑ < 6) {
        _charityFee = charityFee↑;
    }
}

ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner {
    _taxFee = 0;
    if(taxFee↑ < 6) {
        _taxFee = taxFee↑;
    }
}

ftrace | funcSig
function setMarketingDevFeePercent(uint256 marketingAndDevBudget↑) external onlyOwner {
    _marketingAndDevBudget = 0;
    if(marketingAndDevBudget↑ < 6) {
        _marketingAndDevBudget = marketingAndDevBudget↑;
    }
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner {
    _liquidityFee = 0;
    if(liquidityFee↑ < 6) {
        _liquidityFee = liquidityFee↑;
    }
}
```

- The owner can set a max tax percentage.



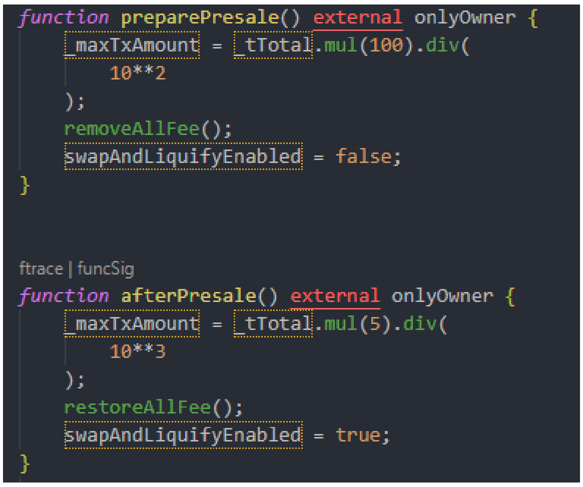
```
function setMaxTxPercent(uint256 maxTxPercent↑) external onlyOwner {
    _maxTxAmount = _tTotal.mul(maxTxPercent↑).div(
        10**3
    );
}
```

- The owner can set swap and liquify enable status.



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

- The owner of the contract can start and complete the presale. Using these functions the owner could set f swap and liquify enable status, and a max tax percentage.



```
function preparePresale() external onlyOwner {
    _maxTxAmount = _tTotal.mul(100).div(
        10**2
    );
    removeAllFee();
    swapAndLiquifyEnabled = false;
}


ftrace | funcSig
function afterPresale() external onlyOwner {
    _maxTxAmount = _tTotal.mul(5).div(
        10**3
    );
    restoreAllFee();
    swapAndLiquifyEnabled = true;
}
```

5 Audit Result

LEVEL	ISSUES
Weakness	DoS with Block Gas Limit (2)
Low	Owner Privilegies (7)

1. The contract utilizes SafeMath libraries along with following the ERC20 standard.
2. There is a 'Marketing fee', a 'Liquidity fee' and a ‘Tax Fee’ on all transactions for any non-excluded address that participates in a transfer. The owner can update the tax rates at any time, though they are limited to 6% for each tax.
3. The owner can also exclude and include users from the fee mechanism.
4. There is a transfer limit on the number of tokens which can be sent in a single transaciton which can be updated by the owner of the contract. The owner can also lock tokens in any address.
5. Some functions could have been declared external instead of public to save some gas, but as this is already deployed this is merely informational.
6. Users who hold tokens will automatically receive a portion the fees from a transaction tax on each transfer.
7. Another portion of the transaction tax will be sent to a charity wallet controlled by the team.
8. A third portion of the transaction tax will be sent to a marketingdev wallet controlled by the team.

9. The final portion of the fee charged on transactions is stored in the contract and, once a threshold value is met, these tokens will be sold for BNB.

10. The resulting BNB is paired with tokens collected to add liquidity.

11. The LP tokens from this process will go to the charity wallet, both controlled by the team.

5.1 Findings Summary

🟩

LittleFlokiFrunk
Low Risk Level

- ✓ **No external vulnerabilities were identified within the smart contract's code**
- ✓ **The code is fully customized**
- ✓ **As with any presale, ensure trust in the team prior to investing**
- ✓ **Ensure trust in the team as they have substantial control over the ecosystem and will control the charity/marketing wallets**
- ✓ **LittleFlokiFrunk token was audited, and no issues were found**

6 Disclamer

CheckPoint team issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these. For the facts that occurred or existed after the issuance, CheckPoint is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to CheckPoint by the information provider till the date of the insurance report. CheckPoint is not responsible for the background and other conditions of the project.

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), CheckPoint suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

CheckPoint

Website

https://checkpoint.report

E-mail

contact@checkpoint.report

Telegram

@checkpointreport

Github

https://github.com/checkpointreport