CheckPoint

Token Security Audit Report
Prepared for MiniRox

[v.1.0]

September 2021

Document Properties

Client	MiniRox
Platform	Binance Smart Chain
Language	Solidity
Codebase	0xD70e14A878E068ee81e412ff88b9131d915bAb8e

Audit Summary

Delivery Date	03.09.2021
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 03/09/2021, CheckPoint conducted a full audit for the MiniRox 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.

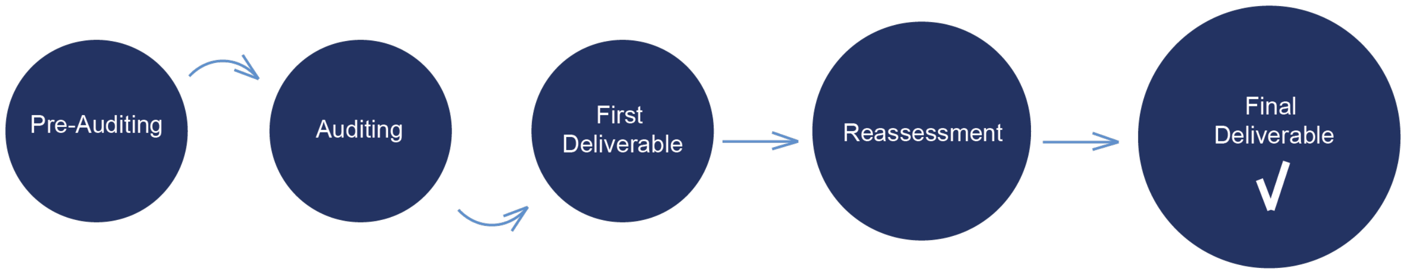
After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to. More information can be found in **Section 5 'Audit Result'**. Practical recommendations are provided according to each vulnerability found and should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

🟨 **MiniRox**
Medium 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**

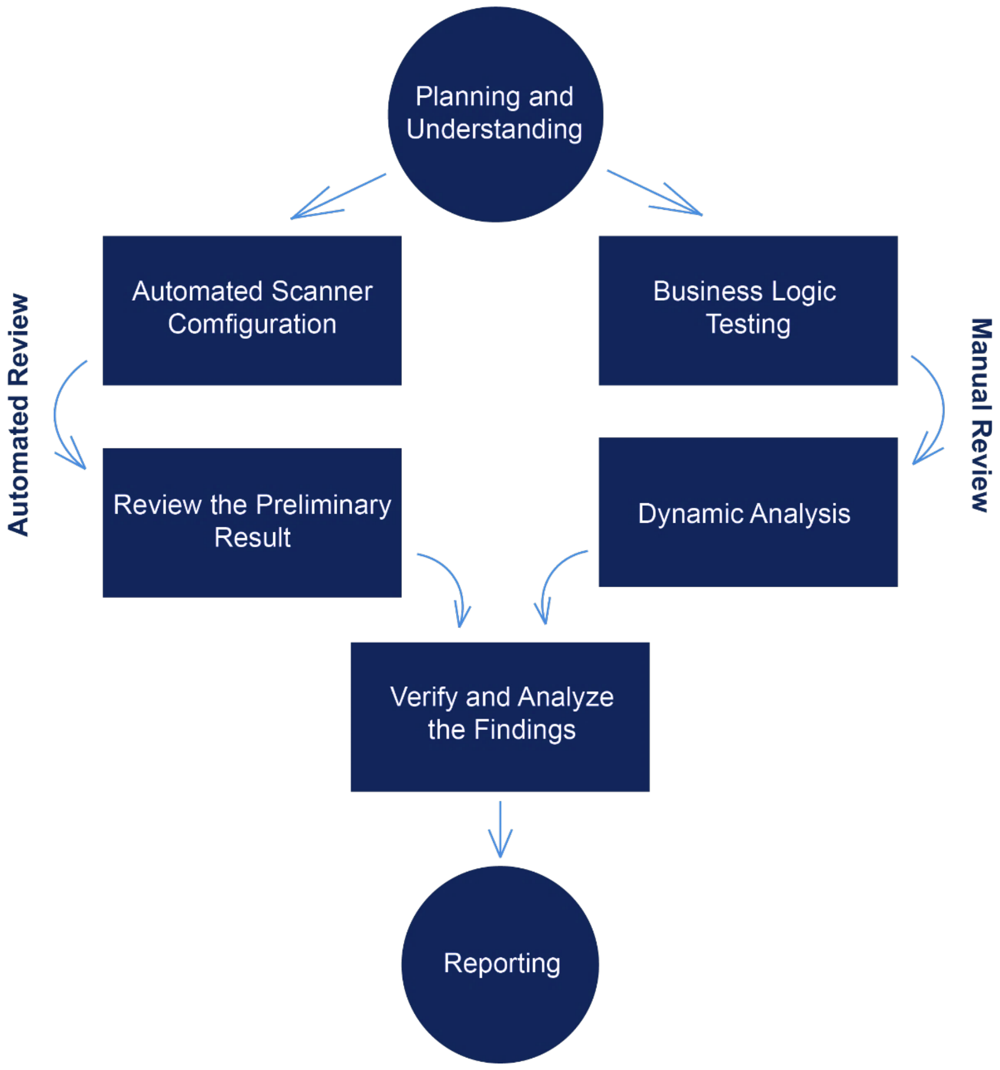
  Providing a full report with the detailed status of each issue.

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 Review
Automated Scanner Comfiguration
Review the Preliminary Result
Business Logic Testing
Dynamic Analysis
Manual Review
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	Weakness	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical
	Low	*Medium*	*High*
		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

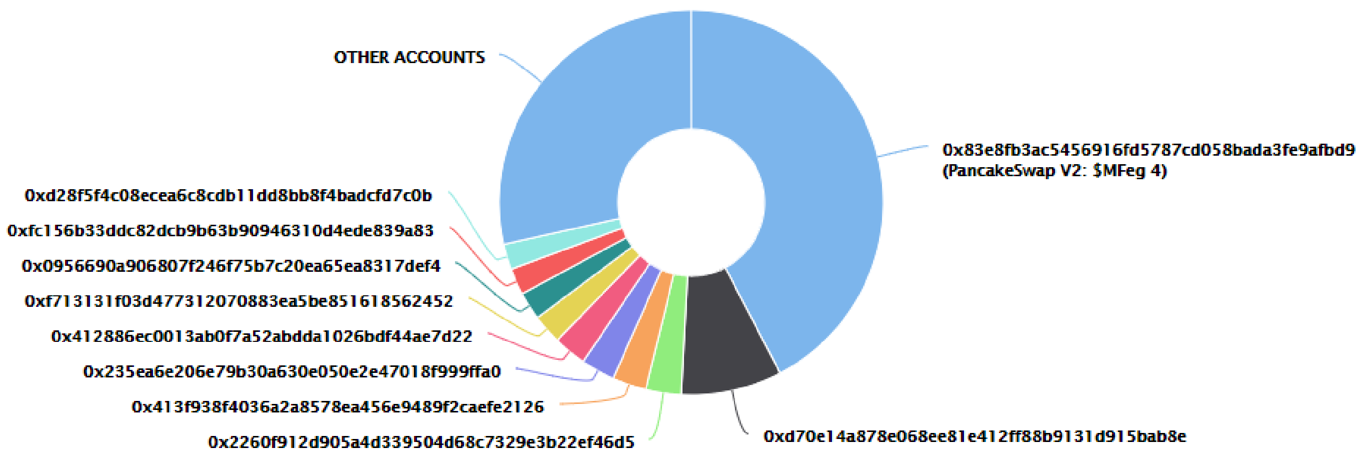
- ✓ **No Website [RISK]**
- ✓ **Only 1 Social Media Networks [RISK]**
- ✓ **< 1000 Telegram Members [RISK]**
- ✓ **Active Voice Chats**
- ✓ **Spam Detected [RISK]**

Remark: This page contains active link

4.2 Smart Contract Details

Contract Name	MiniRox
Contract Address	0xD70e14A878E068ee81e412ff88b9131d915bAb8e
Total Supply	1,000,000,000
Token Ticker	MFeg
Decimals	4
Token Holders	76
Transactions Count	580
Top 100 Holders Dominance	100%
Liquidity Fee	3%
Reflection Fee	4 %
Marketing Fee	3%
Contract Deployer Address	0xa11868c0A72CED371EF37120bfb8ed67E9E2dB95
Current Owner Address	0xa11868c0a72ced371ef37120bfb8ed67e9e2db95

MiniRox Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: $MFeg 4	423,411,775.061	42.3412%
2	0xd70e14a878e068ee81e412ff88b9131d915bab8e	85,115,932.9896	8.5116%
3	0x2260f912d905a4d339504d68c7329e3b22ef46d5	29,580,441.4969	2.9580%
4	0x413f938f4036a2a8578ea456e9489f2caefe2126	28,564,830.4946	2.8565%
5	0x235ea6e206e79b30a630e050e2e47018f999ffa0	28,490,620.2599	2.8491%
6	0x412886ec0013ab0f7a52abdda1026bdf44ae7d22	27,968,341.9733	2.7968%
7	0xf713131f03d477312070883ea5be851618562452	24,791,923.7983	2.4792%
8	0x0956690a906807f246f75b7c20ea65ea8317def4	23,539,431.1883	2.3539%
9	0xfc156b33ddc82dcb9b63b90946310d4ede839a83	22,223,700	2.2224%
10	0xd28f5f4c08ecea6c8cdb11dd8bb8f4badcfd7c0b	21,508,980.4698	2.1509%

4.3 Contract Function Details

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

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

+ [Int] IBEP20
   - [Ext] totalSupply
   - [Ext] decimals
   - [Ext] symbol
   - [Ext] name
   - [Ext] getOwner
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ Auth
   - [Pub] #
   - [Pub] authorize #
      - modifiers: onlyOwner
   - [Pub] unauthorize #
      - modifiers: onlyOwner
   - [Pub] isOwner
   - [Pub] isAuthorized
   - [Pub] transferOwnership #
      - modifiers: onlyOwner

+ [Int] IDEXFactory
   - [Ext] createPair #

+ [Int] IDEXRouter
   - [Ext] factory
   - [Ext] WETH
   - [Ext] addLiquidity #
   - [Ext] addLiquidityETH $
   - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
   - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens $
   - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

 + [Int] IDividendDistributor
   - [Ext] setDistributionCriteria #
   - [Ext] setShare #
   - [Ext] deposit $
   - [Ext] process #

 + DividendDistributor (IDividendDistributor)
   - [Pub] <Constructor> #
   - [Ext] setDistributionCriteria #
      - modifiers: onlyToken
   - [Ext] setShare #
      - modifiers: onlyToken
   - [Ext] deposit $
      - modifiers: onlyToken
   - [Ext] process #
      - modifiers: onlyToken
   - [Int] shouldDistribute
   - [Int] distributeDividend #
   - [Ext] claimDividend #
   - [Pub] getUnpaidEarnings
   - [Int] getCumulativeDividends
   - [Int] addShareholder #
   - [Int] removeShareholder #

 + MiniRox (IBEP20, Auth)
   - [Pub] <Constructor> #
      - modifiers: Auth
   - [Ext] <Fallback> $
   -[Ext] <Fallback> $
   - [Ext] totalSupply
   - [Ext] decimals
   - [Ext] symbol
   - [Ext] name
   - [Ext] getOwner
   - [Pub] balanceOf
   - [Ext] allowance
   - [Pub] approve #
   - [Ext] approveMax #
   - [Ext] transfer #
   - [Ext] transferFrom #
   - [Ext] setMaxWallet #
      - modifiers: onlyOwner
   - [Int] transferFrom #
   - [Int] _basicTransfer #
   - [Int] checkTxLimit
   - [Int] shouldTakeFee
   - [Int] takeFee #
 - [Int] shouldSwapBack
 - [Ext] clearStuckBalance #
   - modifiers: onlyOwner
 - [Pub] tradingStatus #
   - modifiers: onlyOwner
 - [Pub] cooldownEnabled #
   - modifiers: onlyOwner
 - [Int] swapBack #
   - modifiers: swapping
 - [Ext] setTxLimit #
   - modifiers: authorized
 - [Ext] setIsDividendExempt #
   - modifiers: authorized
 - [Ext] setIsFeeExempt #
   - modifiers: authorized
 - [Ext] setIsTxLimitExempt #
   - modifiers: authorized
 - [Ext] setIsTimelockExempt #
   - modifiers: authorized
 - [Ext] setFees #
   - modifiers: onlyOwner
 - [Ext] setFeeReceivers #
   - modifiers: onlyOwner
 - [Ext] setSwapBackSettings #
   - modifiers: authorized
 - [Ext] setTargetLiquidity #
   - modifiers: authorized
 - [Ext] setDistributionCriteria #
   - modifiers: authorized
 - [Ext] setDistributorSettings #
   - modifiers: authorized
 - [Pub] getCirculatingSupply
 - [Pub] getLiquidityBacking
 - [Pub] isOverLiquified
 - [Ext] airdrop #
   - modifiers: onlyOwner
```

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	LOW RISK
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

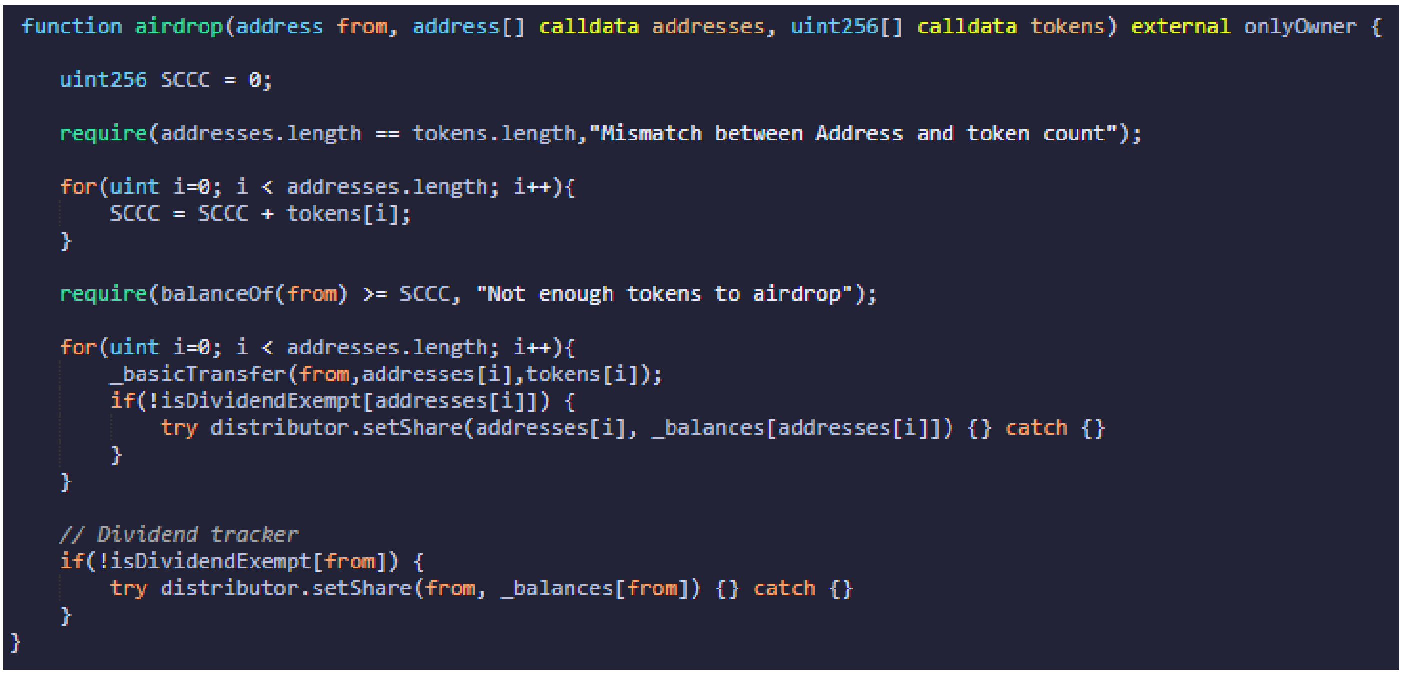
CHECKING ITEM	NOTES	RESULT
Right-To-Left-Override Control Character	N / A	PASS
Shadowing State Variables	N / A	PASS
Signature Malleability	N / A	PASS
State Variable Default Visibility	N / A	PASS
Timestamp Dependence	N / A	PASS
Transaction Order Dependence	N / A	PASS
Typographical Error	N / A	PASS
Unencrypted 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 airdrop() uses the loop to airdrop rewords by the list. This function will be aborted with out-of-gas exception if there will be a long excluded addresses list.

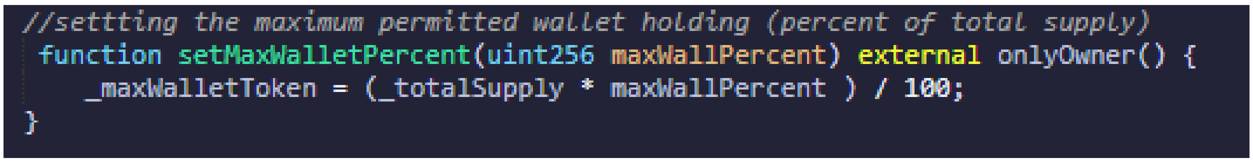


```
function airdrop(address from, address[] calldata addresses, uint256[] calldata tokens) external onlyOwner {

    uint256 SCCC = 0;

    require(addresses.length == tokens.length,"Mismatch between Address and token count");

    for(uint i=0; i < addresses.length; i++){
        SCCC = SCCC + tokens[i];
    }

    require(balanceOf(from) >= SCCC, "Not enough tokens to airdrop");

    for(uint i=0; i < addresses.length; i++){
        _basicTransfer(from,addresses[i],tokens[i]);
        if(!isDividendExempt[addresses[i]]) {
            try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
        }
    }

    // Dividend tracker
    if(!isDividendExempt[from]) {
        try distributor.setShare(from, _balances[from]) {} catch {}
    }
}
```

Recommendation: Make sure the length of the excluded array is not too long

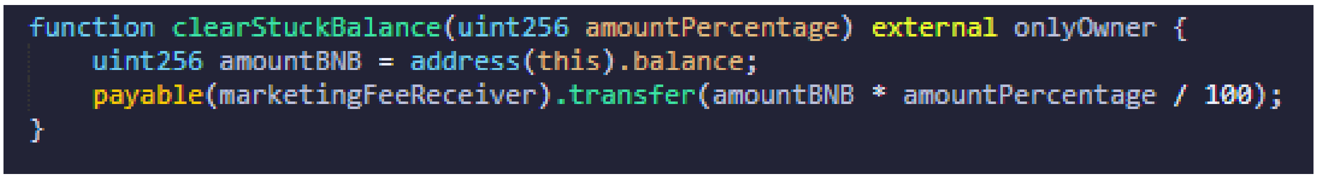
[RISK] Owner Privileges (in the period when the owner is not renounced)

- Owner can set arbitrary _maxWalletAmount.



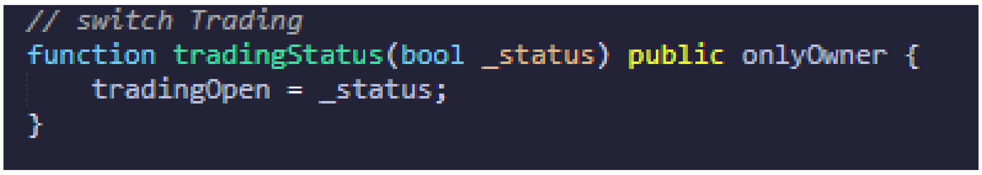
```
//setting the maximum permitted wallet holding (percent of total supply)
 function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner() {
     _maxWalletToken = (_totalSupply * maxWallPercent ) / 100;
 }
```

- Owner can clear StuckBalance.



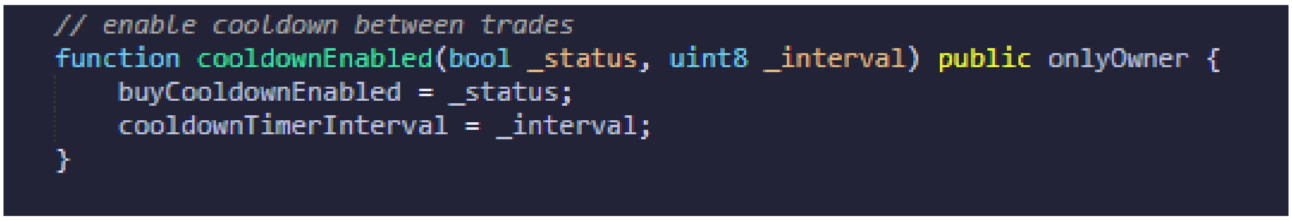
```
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(marketingFeeReceiver).transfer(amountBNB * amountPercentage / 100);
}
```

- Owner can change trading status.



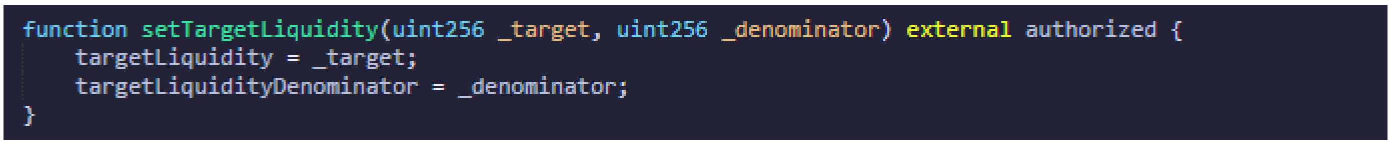
```
// switch Trading
function tradingStatus(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

- Owner can change cooldown status.



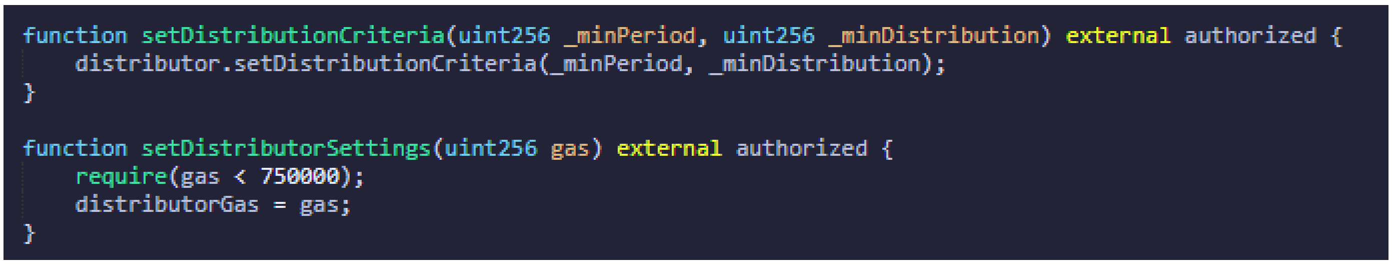
```
  // enable cooldown between trades
  function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
      buyCooldownEnabled = _status;
      cooldownTimerInterval = _interval;
  }
```

- Owner can change target liquidity values.



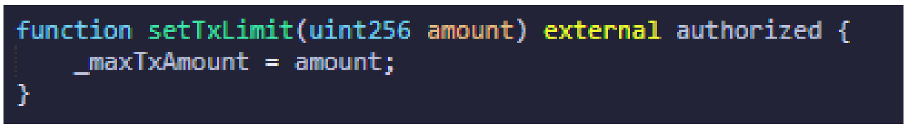
```
function setTargetLiquidity(uint256 _target, uint256 _denominator) external authorized {
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
}
```

- Owner can change distribution criteria and distribution GAS.



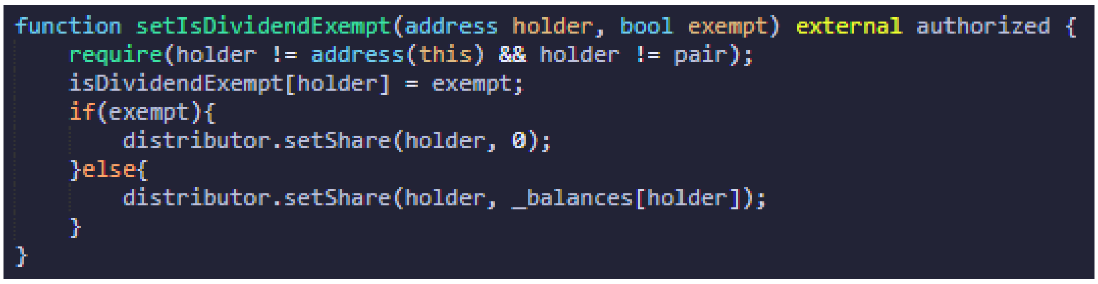
```
function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution) external authorized {
    distributor.setDistributionCriteria(_minPeriod, _minDistribution);
}

function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 750000);
    distributorGas = gas;
}
```

- Owner can change the maximum transaction amount.



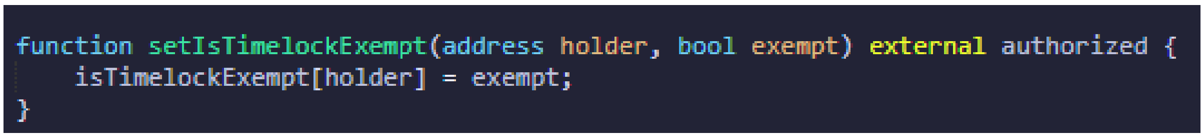
```
function setTxLimit(uint256 amount) external authorized {
    _maxTxAmount = amount;
}
```

- Owner can include in and exclude from dividends.



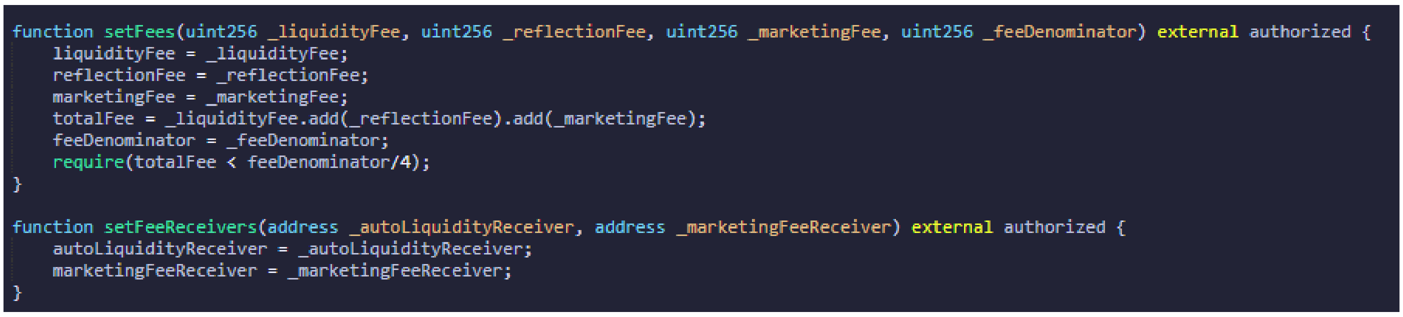
```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
        distributor.setShare(holder, 0);
    }else{
        distributor.setShare(holder, _balances[holder]);
    }
}
```

- Owner can change addresses’ isTimelockExempt value.



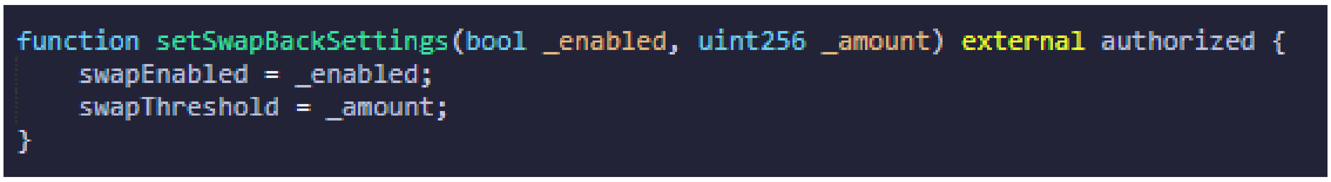
```
function setIsTimelockExempt(address holder, bool exempt) external authorized {
    isTimelockExempt[holder] = exempt;
}
```

- Owner can change fees and fee receivers.



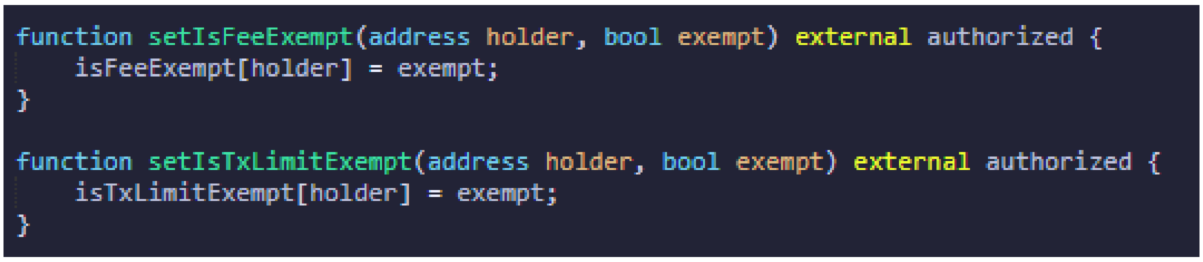
```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/4);
}

function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
}
```

- Owner can change swap threshold and disable/enable swap.



```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

- Owner can include in and exclude from fee and transaction amount.



```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}

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

5 Audit Result

LEVEL	ISSUES
Weakness	DoS with Block Gas Limit (1)
Medium	Owner Privilegies (12)

1. The contract utilizes SafeMath libraries along with following the ERC20 standard.

2. The owner has the ability to set and update a maximum transaction percent at any time, which will impose a limit to the number of tokens that can be transferred during any given transaction. Maximum value not set. **[RISK]**

3. There is a 'liquidity fee', a ‘marketing fee’ and a ‘reflection fee’ on all transactions for any address that participates in a transfer. The owner has the ability to modify these fees from 10 to 100 percent, at any time. Maximum value not set. **[RISK]**

4. A portion of the tax fee is redistributed to existing token holders instantly and automatically at the time of each transaction.

5. The owner of the contract can exclude accounts from transfer fees and reward distribution.

6. Ownership has not been renounced.

5.1 Findings Summary

MiniRox

Medium Risk Level

- ✓ **No external vulnerabilities were identified within the smart contract's code**
- ✓ **We strongly recommend that the team renounces ownership**
- ✓ **Please ensure trust in the team prior to investing as they have substantial control within the ecosystem**
- ✓ **We strongly recommend that the contract owners remove errors and re-audit**

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