RugFreeCoins Audit

Rainbow Farm Token

Smart Contract Security Audit

November 01, 2021

Contents

Audit details

Audited project

Rainbow farm Token

Contract Address

RNBO MC:

0xf0a0E6Cc29Fd225E7ED8C4D983f1Bb0d874BB812

RNBZ:

0xE3db2FA2900072e4eE2215f5c25e603de0f8FdB9

RNBZ MC:

0xDE6b77E0Ae277805135B4372c4A8Acc9A07ae0A2

Client contact

Rainbow farm Team

Blockchain

Binance smart chain

Project website

https://rainbowfarm.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 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 Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins 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 Rugfreecoins 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, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, 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

Rugfreecoins was commissioned by Rainbow Farm Token to perform an audit of the smart contract.

https://bscscan.com/address/0xf0a0E6Cc29Fd225E7ED8C4D983f1Bb0d874BB812

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

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

About the project

Rainbow Farm is a farming token built on the Binance Smart Chain.

The project has an existing token called RNBO and investors will stake RNBO token and LP to earn RMBZ that is through RNBZ MC. There is no other way to earn RNBZ and liquidity will be added after 15 days for RNBZ so the only way to get RNBZ would be through RNBO. In normal MC people can stake only if they hold RNBZ and stake it. There will be a minimum of 1000 RNBZ required to enter normal pools and farms.

They have an APR boost system that depends on how long people stay in without withdrawal and harvesting. After 1st week each day adds 1% of boost and the highest boost is 100%. Similarly, withdrawal fees post 45 days are half of the actual fees, and after 90 days no withdrawal fees. In the first week after depositing there will be a 10% penalty on the harvest.

On withdrawal, users need to have minimum required RNBZ then there will be 10% fees also while withdrawing RNBZ if the user is staking in any other pool or farm then they can withdraw all except the minimum required to stay in other farms and pools.

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	9/10
3	Information quality	6/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	8/10
Total Points		**8.125/10**

Contract details

Token contract details for 01st November 2021

Contract name	RNBOExclusiveMC
Contract address	0xf0a0E6Cc29Fd225E7ED8C4D983f1Bb0d874BB812
Dev & marketing wallet	0xdbc22cdd2717bc6fd3faa3c7956cf1154222e942
Fee address	0xec50cd98a33542ffb5d1182a55c49402769ba983
Contract deployer address	0xa25A93F5029e14b8a26b871f353Fc9d0762C04Ca
Contract’s current owner address	0x1f4f94db73b235287661583456381b8ff80d9460

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	low issue
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

Contract description table

Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.

RNBOExclusiveMC

Contract	**Type**	**Bases**		
└	Function Name	Visibility	Mutability	Modifiers
ReentrancyGuard	**Implementation**			
└		Public ❗	🛑	NO❗
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
IERC20	**Interface**			

└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeERC20	**Library**			
└	safeTransfer	Internal 🔒	🛑	

L	safeTransferFrom	Internal 🔒	🛑	
L	safeApprove	Internal 🔒	🛑	
L	safeIncreaseAllowance	Internal 🔒	🛑	
L	safeDecreaseAllowance	Internal 🔒	🛑	
L	_callOptionalReturn	Private 🔐	🛑	
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Internal 🔒	🛑	
L	owner	Public ❗️		NO❗️
L	renounceOwnership	Public ❗️	🛑	onlyOwner
L	transferOwnership	Public ❗️	🛑	onlyOwner
Roles	**Library**			
L	add	Internal 🔒	🛑	
L	remove	Internal 🔒	🛑	
L	has	Internal 🔒		

MinterRole	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	isMinter	Public ❗		NO❗
└	addMinter	Public ❗	🛑	onlyMinter
└	renounceMinter	Public ❗	🛑	NO❗
└	_addMinter	Internal 🔒	🛑	
└	_removeMinter	Internal 🔒	🛑	
RNBZ	**Implementation**	**Context, IERC20, Ownable, MinterRole**		
└		Public ❗	🛑	NO❗
└	mint	Public ❗	🛑	onlyMinter
└	burn	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗

└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_getCurrentSupply	Private 🔐		
└	safe32	Internal 🔒		
└	getChainId	Internal 🔒	🛑	
RainbowToken	**Implementation**	**Context, IERC20, Ownable, MinterRole**		
└		Public ❗	🛑	NO❗
└	mint	Public ❗	🛑	onlyMinter
└	burn	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗

└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	isExcluded	Public ❗		NO❗
└	excludeAccount	External ❗	🛑	onlyOwner
└	includeAccount	External ❗	🛑	onlyOwner
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_getCurrentSupply	Private 🔐		
└	delegates	External ❗		NO❗
└	delegate	External ❗	🛑	NO❗
└	delegateBySig	External ❗	🛑	NO❗

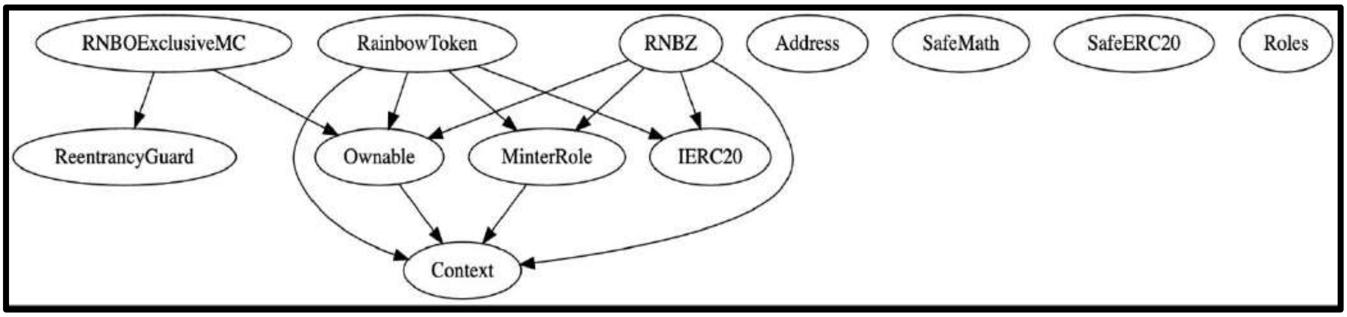
└	getCurrentVotes	External ❗		NO❗
└	getPriorVotes	External ❗		NO❗
└	_delegate	Internal 🔒	🛑	
└	_moveDelegates	Internal 🔒	🛑	
└	_writeCheckpoint	Internal 🔒	🛑	
└	safe32	Internal 🔒		
└	getChainId	Internal 🔒	🛑	
RNBOExclusiveMC	**Implementation**	**Ownable, ReentrancyGuard**		
└		Public ❗	🛑	NO❗
└	poolLength	External ❗		NO❗
└	setMaxWithdrawFee	Public ❗	🛑	onlyOwner
└	setMinRNBZHolding	Public ❗	🛑	onlyOwner
└	add	External ❗	🛑	onlyOwner nonDuplicated
└	set	External ❗	🛑	onlyOwner
└	getMultiplier	Public ❗		NO❗
└	pendingRNBO	External ❗		NO❗
└	hasBalanceStaked	Internal 🔒		

L	massUpdatePools	Public ❗️	🛑	NO❗️
L	updatePool	Public ❗️	🛑	NO❗️
L	deposit	Public ❗️	🛑	nonReentrant
L	getActualWithdrawFeeRate	Public ❗️		NO❗️
L	getBoostRewardRate	Public ❗️		NO❗️
L	withdraw	Public ❗️	🛑	nonReentrant
L	emergencyWithdraw	Public ❗️	🛑	nonReentrant
L	safeRNBOTransfer	Internal 🔒	🛑	
L	setDevAddress	External ❗️	🛑	onlyOwner
L	setFeeAddress	External ❗️	🛑	onlyOwner
L	updateEmissionRate	External ❗️	🛑	onlyOwner
L	updateStartBlock	External ❗️	🛑	onlyOwner

Legend

Symbol	**Meaning**
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy



RNBZ TOKEN

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	

└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		

Ownable	**Implementation**	**Context**		
└		Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Roles	**Library**			
└	add	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
└	has	Internal 🔒		
MinterRole	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	isMinter	Public ❗		NO❗
└	addMinter	Public ❗	🛑	onlyMinter
└	renounceMinter	Public ❗	🛑	NO❗
└	_addMinter	Internal 🔒	🛑	
└	_removeMinter	Internal 🔒	🛑	

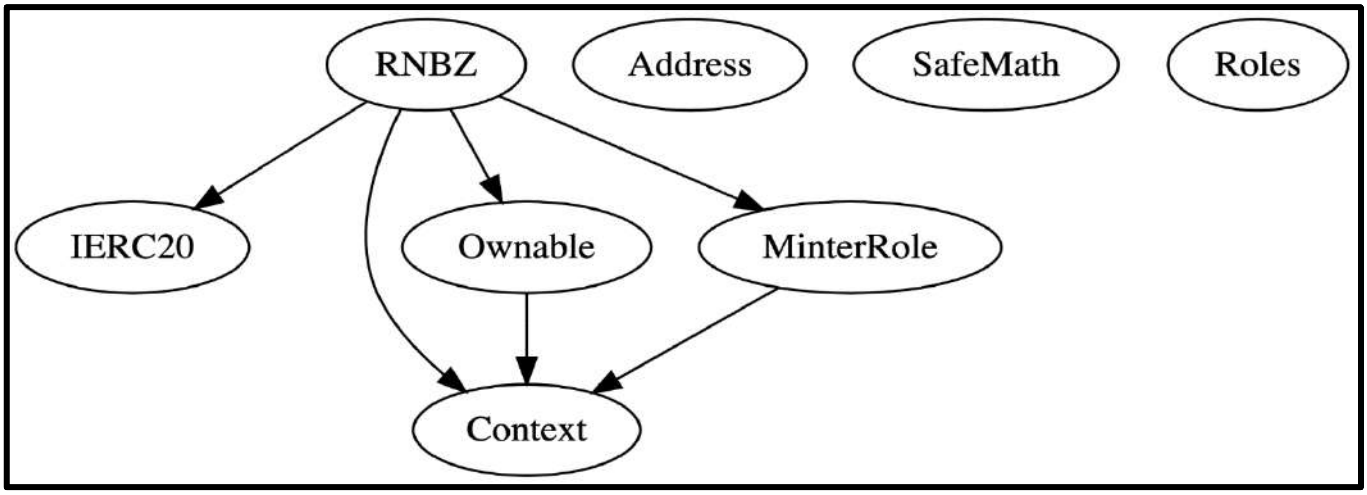
RNBZ	Implementation	Context, IERC20, Ownable, MinterRole		
└		Public ❗	🛑	NO❗
└	mint	Public ❗	🛑	onlyMinter
└	burn	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_getCurrentSupply	Private 🔐		
└	safe32	Internal 🔒		

└	getChainId	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy



RNBZ_MasterChef

Contract	**Type**	**Bases**		
└	Function Name	Visibility	Mutability	Modifiers
ReentrancyGuard	**Implementation**			
└		Public ❗	🛑	NO❗
Address	**Library**			
└	isContract	Internal 🔒		

└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		

└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeERC20	**Library**			
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeApprove	Internal 🔒	🛑	
└	safeIncreaseAllowance	Internal 🔒	🛑	
└	safeDecreaseAllowance	Internal 🔒	🛑	
└	_callOptionalReturn	Private 🔐	🛑	
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		

└		Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Roles	**Library**			
└	add	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
└	has	Internal 🔒		
MinterRole	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	isMinter	Public ❗		NO❗
└	addMinter	Public ❗	🛑	onlyMinter
└	renounceMinter	Public ❗	🛑	NO❗
└	_addMinter	Internal 🔒	🛑	
└	_removeMinter	Internal 🔒	🛑	
RNBZ	**Implementation**	**Context, IERC20, Ownable, MinterRole**		

└		Public ❗️	🛑	NO❗️
└	mint	Public ❗️	🛑	onlyMinter
└	burn	Public ❗️	🛑	NO❗️
└	name	Public ❗️		NO❗️
└	symbol	Public ❗️		NO❗️
└	decimals	Public ❗️		NO❗️
└	totalSupply	Public ❗️		NO❗️
└	balanceOf	Public ❗️		NO❗️
└	transfer	Public ❗️	🛑	NO❗️
└	allowance	Public ❗️		NO❗️
└	approve	Public ❗️	🛑	NO❗️
└	transferFrom	Public ❗️	🛑	NO❗️
└	increaseAllowance	Public ❗️	🛑	NO❗️
└	decreaseAllowance	Public ❗️	🛑	NO❗️
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_getCurrentSupply	Private 🔐		
└	safe32	Internal 🔒		

└	getChainId	Internal 🔒	🛑	
RainbowToken	**Implementation**	**Context, IERC20, Ownable, MinterRole**		
└		Public ❗️	🛑	NO❗️
└	mint	Public ❗️	🛑	onlyMinter
└	burn	Public ❗️	🛑	NO❗️
└	name	Public ❗️		NO❗️
└	symbol	Public ❗️		NO❗️
└	decimals	Public ❗️		NO❗️
└	totalSupply	Public ❗️		NO❗️
└	balanceOf	Public ❗️		NO❗️
└	transfer	Public ❗️	🛑	NO❗️
└	allowance	Public ❗️		NO❗️
└	approve	Public ❗️	🛑	NO❗️
└	transferFrom	Public ❗️	🛑	NO❗️
└	increaseAllowance	Public ❗️	🛑	NO❗️
└	decreaseAllowance	Public ❗️	🛑	NO❗️
└	isExcluded	Public ❗️		NO❗️

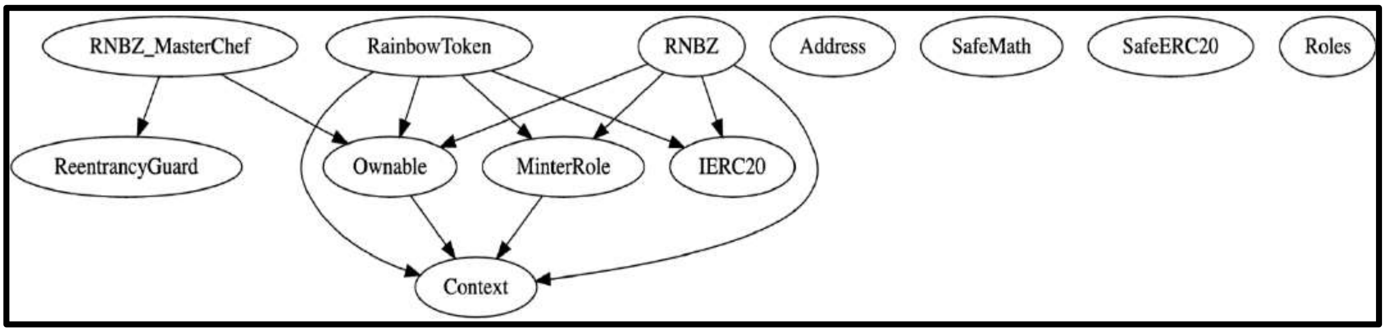
└	excludeAccount	External ❗	🛑	onlyOwner
└	includeAccount	External ❗	🛑	onlyOwner
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_getCurrentSupply	Private 🔐		
└	delegates	External ❗		NO❗
└	delegate	External ❗	🛑	NO❗
└	delegateBySig	External ❗	🛑	NO❗
└	getCurrentVotes	External ❗		NO❗
└	getPriorVotes	External ❗		NO❗
└	_delegate	Internal 🔒	🛑	
└	_moveDelegates	Internal 🔒	🛑	
└	_writeCheckpoint	Internal 🔒	🛑	
└	safe32	Internal 🔒		
└	getChainId	Internal 🔒	🛑	
RNBZ_MasterChef	**Implementation**	**Ownable, ReentrancyGuard**		
└		Public ❗	🛑	NO❗

└	poolLength	External ❗		NO❗
└	setMaxWithdrawFee	Public ❗	🛑	onlyOwner
└	add	External ❗	🛑	onlyOwner nonDuplicated
└	set	External ❗	🛑	onlyOwner
└	getMultiplier	Public ❗		NO❗
└	pendingRNBO	External ❗		NO❗
└	massUpdatePools	Public ❗	🛑	NO❗
└	updatePool	Public ❗	🛑	NO❗
└	deposit	Public ❗	🛑	nonReentrant
└	getActualWithdrawFeeRate	Public ❗		NO❗
└	withdraw	Public ❗	🛑	nonReentrant
└	emergencyWithdraw	Public ❗	🛑	nonReentrant
└	safeRNBZTransfer	Internal 🔒	🛑	
└	setFeeAddress	External ❗	🛑	onlyOwner
└	updateEmissionRate	External ❗	🛑	onlyOwner
└	updateStartBlock	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy

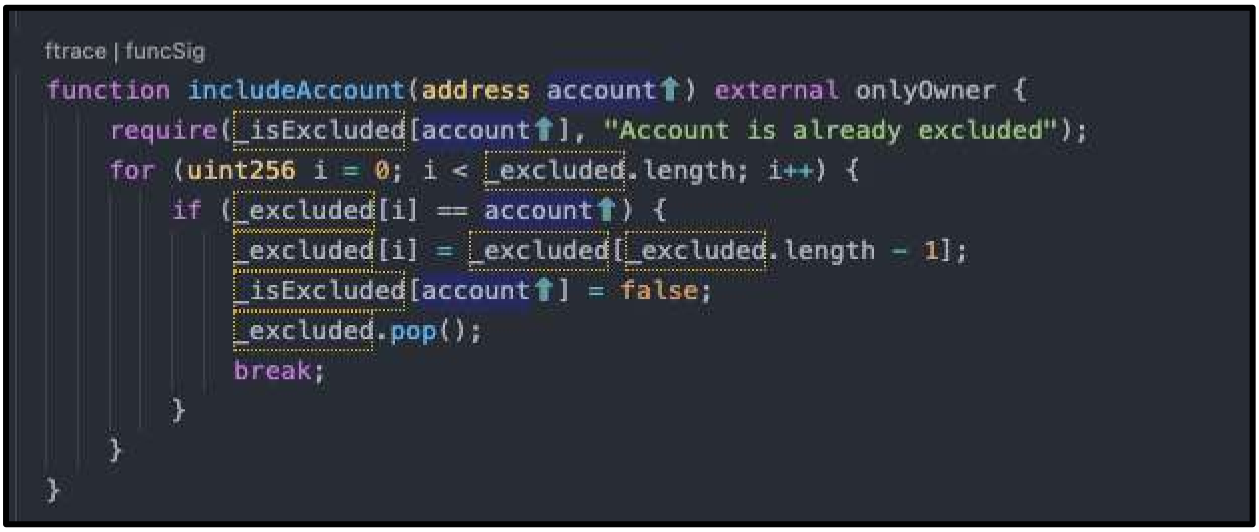


Security issue checking status

- ❖ **High severity issues**
  - **No high severity issues found.**
- ❖ **Medium severity issues**
  - **No medium severity issues found.**

❖ **Low severity issues**

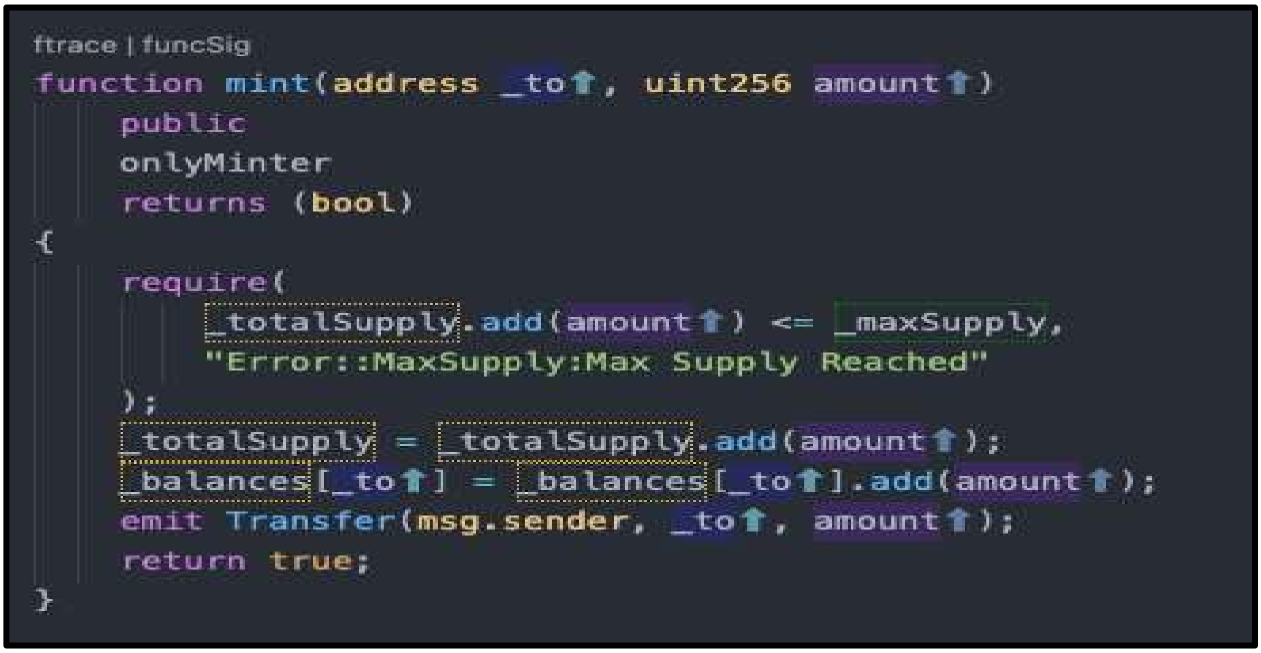
- **In the includeInReward function, if they use a long wallet list there can be an OUT_OF_GAS issue, better to use a small array list at once.**



```
function includeAccount(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];
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

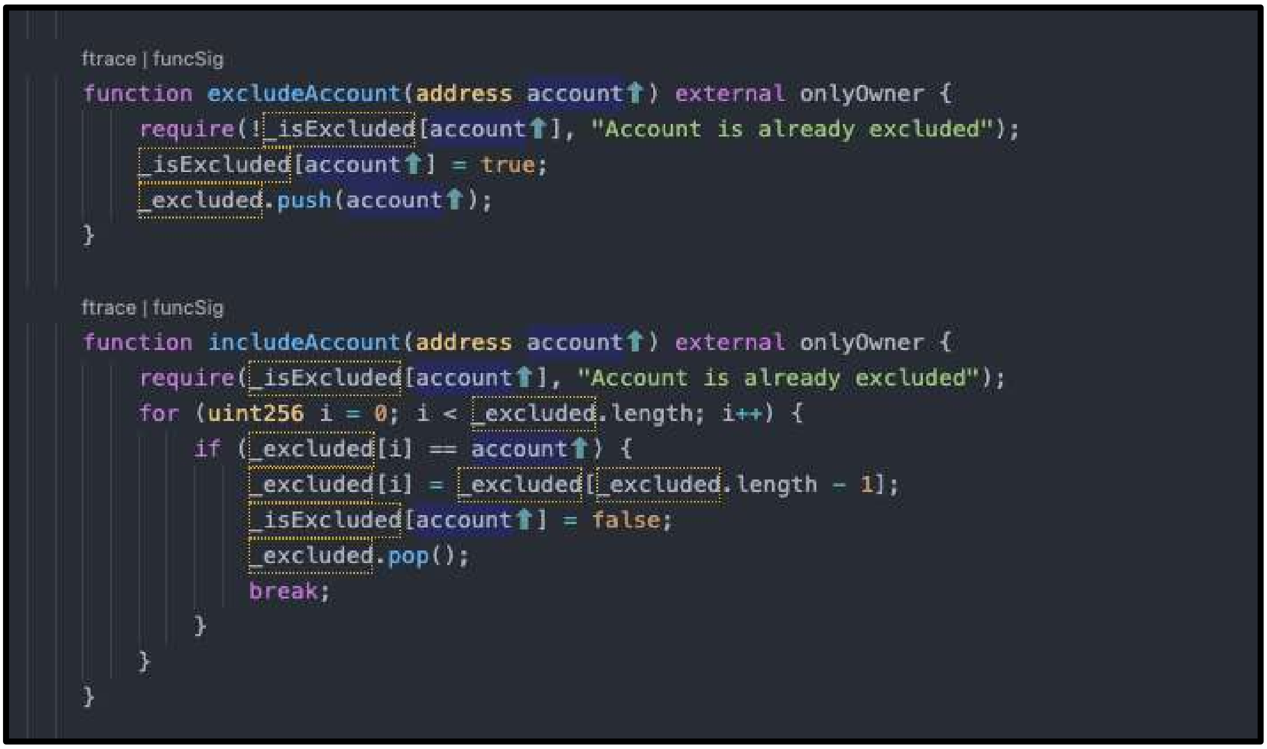
Owner privileges

❖ The owner can mint new tokens.



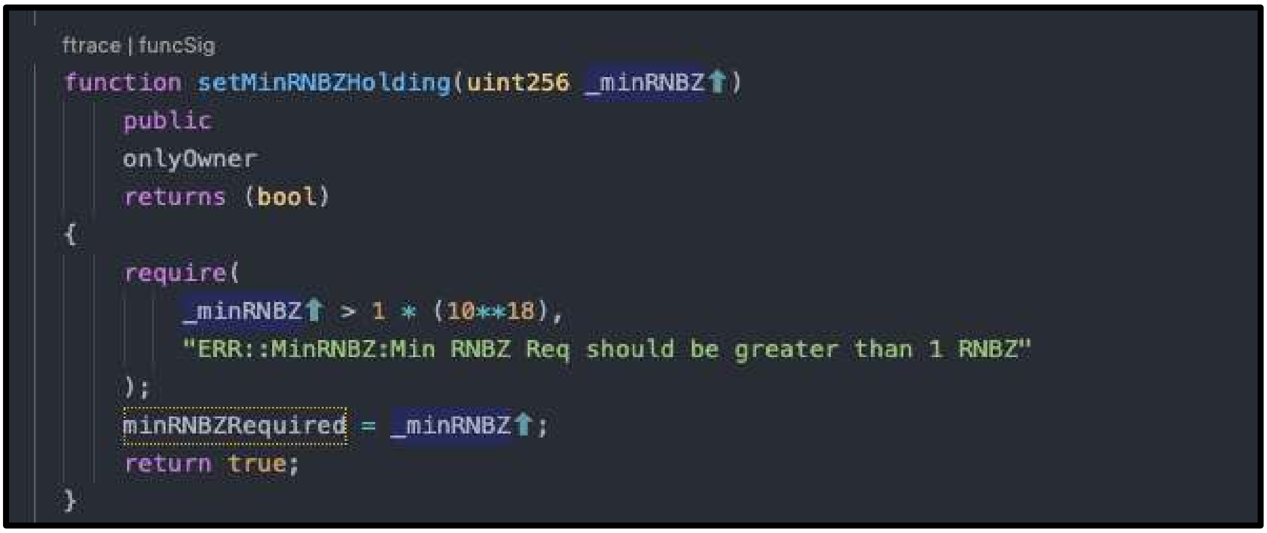
```
function mint(address _to, uint256 amount)
    public
    onlyMinter
    returns (bool)
{
    require(
        _totalSupply.add(amount) <= _maxSupply,
        "Error::MaxSupply:Max Supply Reached"
    );
    _totalSupply = _totalSupply.add(amount);
    _balances[_to] = _balances[_to].add(amount);
    emit Transfer(msg.sender, _to, amount);
    return true;
}
```

- The owner can include and exclude wallets from limitations.



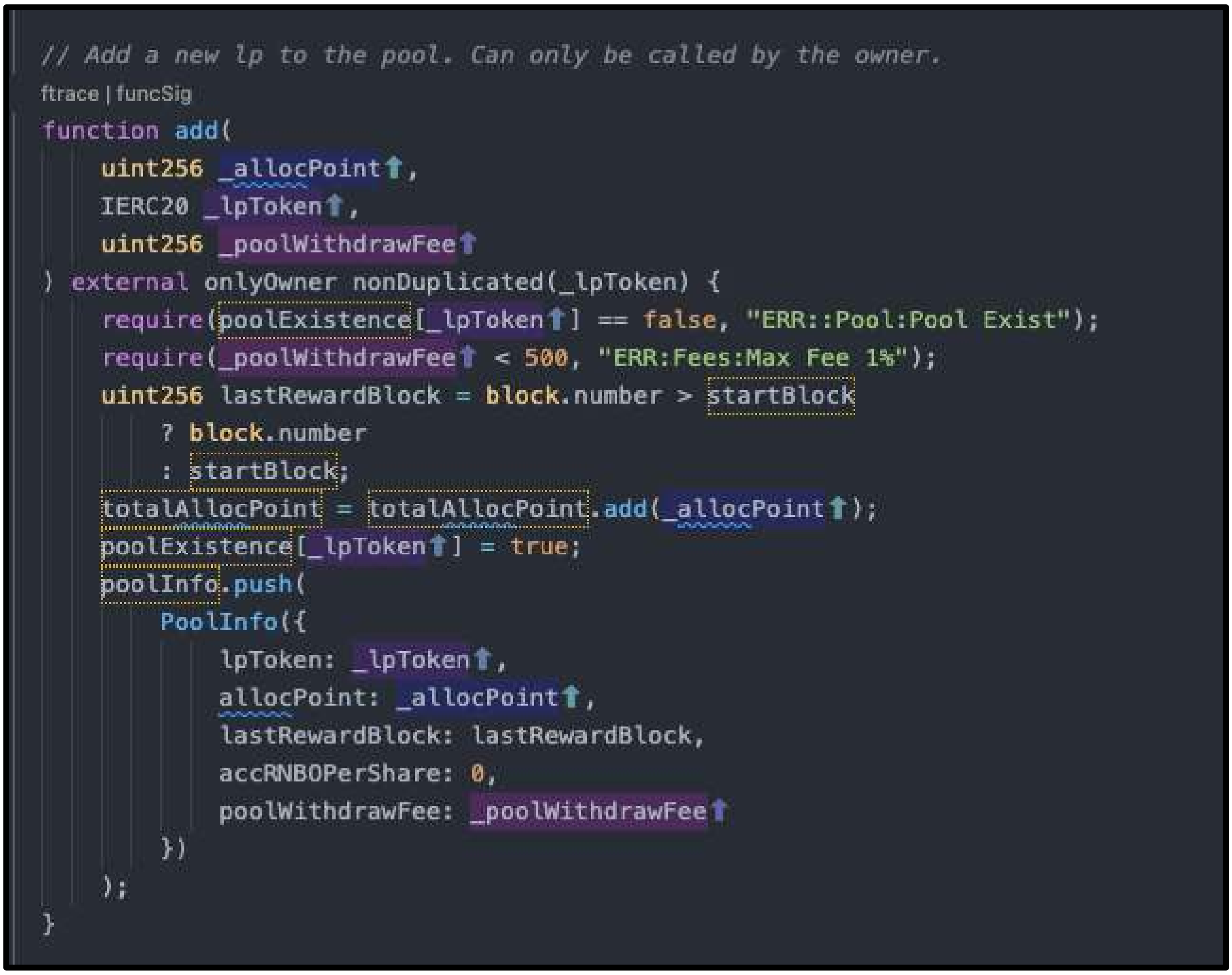
```
ftrace | funcSig
function excludeAccount(address account) external onlyOwner {
    require(!_isExcluded[account], "Account is already excluded");
    _isExcluded[account] = true;
    _excluded.push(account);
}

ftrace | funcSig
function includeAccount(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];
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The owner can change the minimum RNBZ holding amount to Enter pools.



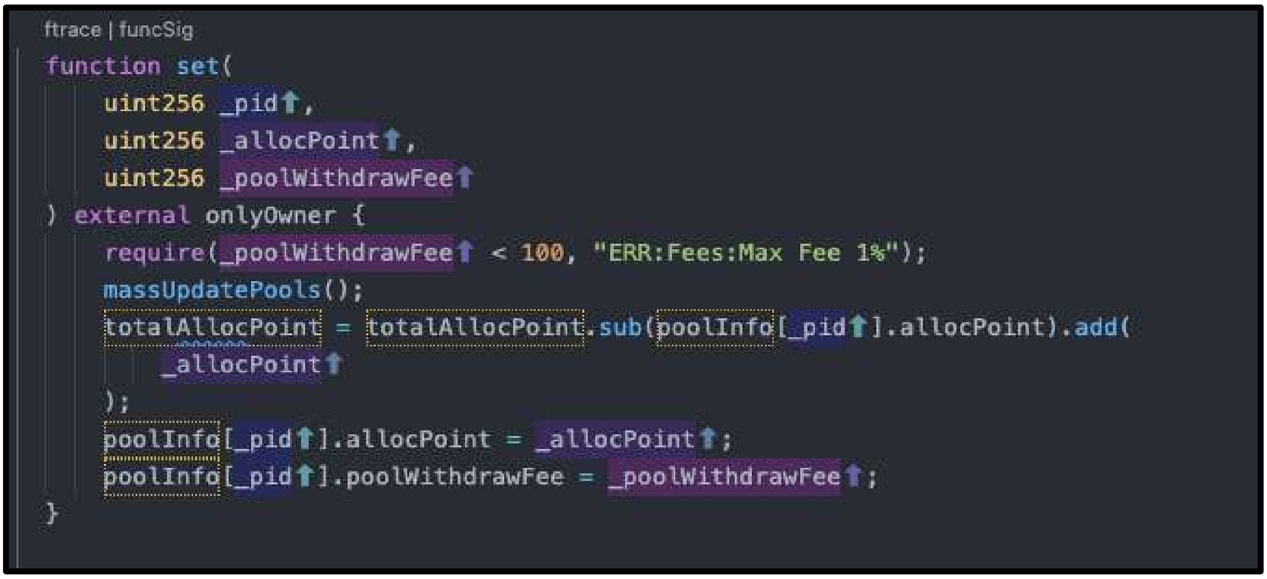
```
ftrace | funcSig
function setMinRNBZHolding(uint256 _minRNBZ)
    public
    onlyOwner
    returns (bool)
{
    require(
        _minRNBZ > 1 * (10**18),
        "ERR::MinRNBZ:Min RNBZ Req should be greater than 1 RNBZ"
    );
    minRNBZRequired = _minRNBZ;
    return true;
}
```

- ❖ The owner can add new pools.



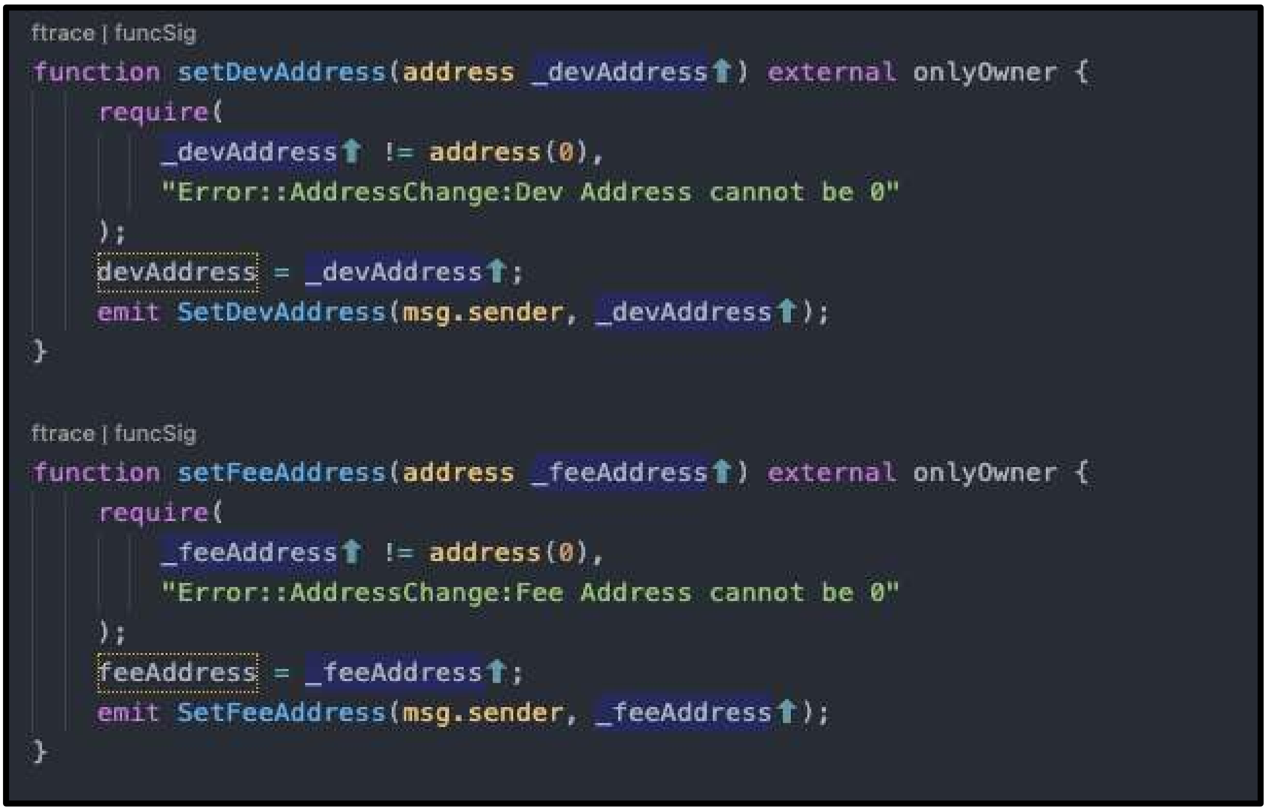
```
// Add a new lp to the pool. Can only be called by the owner.
ftrace | funcSig
function add(
    uint256 _allocPoint,
    IERC20 _lpToken,
    uint256 _poolWithdrawFee
) external onlyOwner nonDuplicated(_lpToken) {
    require(poolExistence[_lpToken] == false, "ERR::Pool:Pool Exist");
    require(_poolWithdrawFee < 500, "ERR:Fees:Max Fee 1%");
    uint256 lastRewardBlock = block.number > startBlock
        ? block.number
        : startBlock;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolExistence[_lpToken] = true;
    poolInfo.push(
        PoolInfo({
            lpToken: _lpToken,
            allocPoint: _allocPoint,
            lastRewardBlock: lastRewardBlock,
            accRNBOPerShare: 0,
            poolWithdrawFee: _poolWithdrawFee
        })
    );
}
```

- ❖ The owner can change the allocated point and withdraw fee (maximum up to 1%) in a pool.



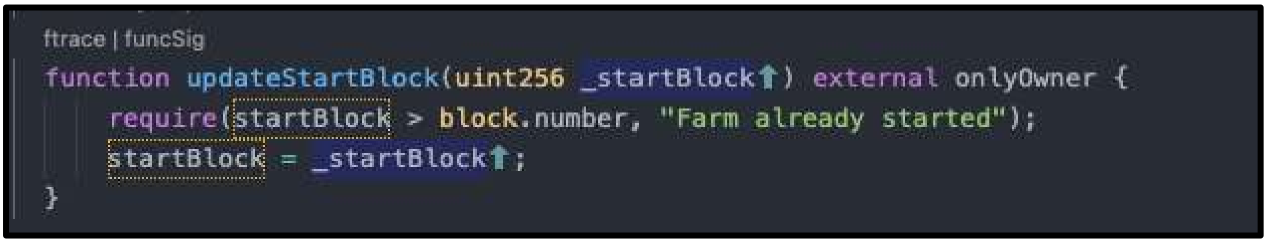
```
ftrace | funcSig
function set(
    uint256 _pid,
    uint256 _allocPoint,
    uint256 _poolWithdrawFee
) external onlyOwner {
    require(_poolWithdrawFee < 100, "ERR:Fees:Max Fee 1%");
    massUpdatePools();
    totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(
        _allocPoint
    );
    poolInfo[_pid].allocPoint = _allocPoint;
    poolInfo[_pid].poolWithdrawFee = _poolWithdrawFee;
}
```

- The owner can change dev and fee address.



```
ftrace | funcSig
function setDevAddress(address _devAddress) external onlyOwner {
    require(
        _devAddress != address(0),
        "Error::AddressChange:Dev Address cannot be 0"
    );
    devAddress = _devAddress;
    emit SetDevAddress(msg.sender, _devAddress);
}

ftrace | funcSig
function setFeeAddress(address _feeAddress) external onlyOwner {
    require(
        _feeAddress != address(0),
        "Error::AddressChange:Fee Address cannot be 0"
    );
    feeAddress = _feeAddress;
    emit SetFeeAddress(msg.sender, _feeAddress);
}
```

- The owner can start farming.



```
ftrace | funcSig
function updateStartBlock(uint256 _startBlock) external onlyOwner {
    require(startBlock > block.number, "Farm already started");
    startBlock = _startBlock;
}
```

Audit conclusion

While conducting the audit of the Rainbow farm smart contract, it was observed that there is nothing alarming with the code and it only contains a low severity issue.