RUGFREECOINS

DripX Miners Token

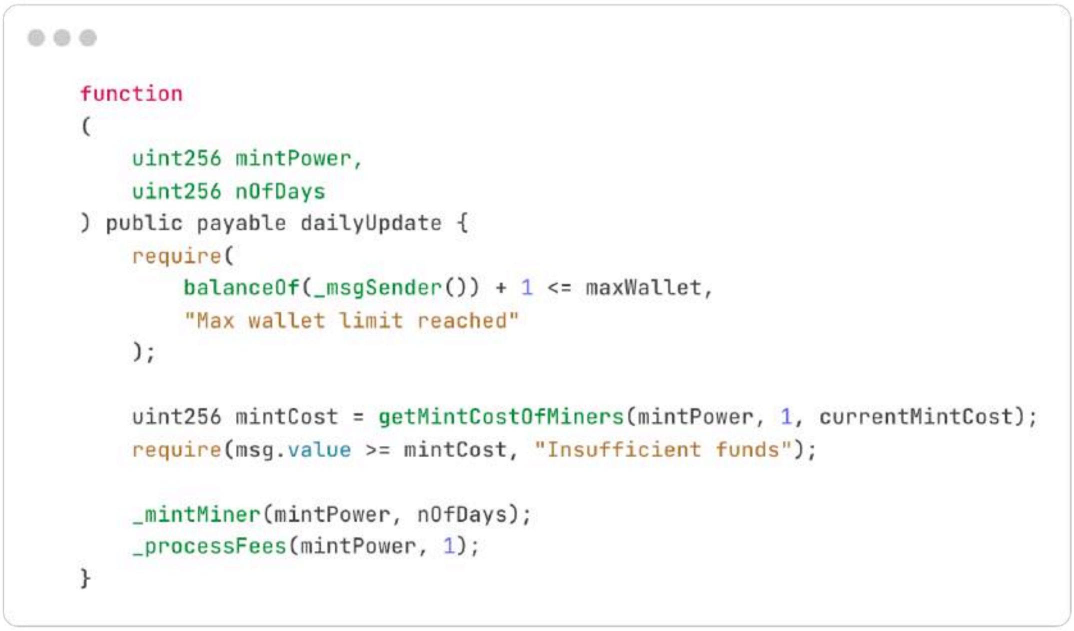
RugfreeCoins Verified on March 07th, 2024

Overview

❗ HIGH SEVERITY ISSUES

The contract is an upgradable contract, the owner can change the functions later

The function on line number 411 lacks a function name; consequently, it will not function properly and will fail to compile. Fixed



```
function
(
    uint256 mintPower,
    uint256 nOfDays
) public payable dailyUpdate {
    require(
        balanceOf(_msgSender()) + 1 <= maxWallet,
        "Max wallet limit reached"
    );

    uint256 mintCost = getMintCostOfMiners(mintPower, 1, currentMintCost);
    require(msg.value >= mintCost, "Insufficient funds");

    _mintMiner(mintPower, nOfDays);
    _processFees(mintPower, 1);
}
```

Contents

Audit details

Audited project

DripX Miners Token

Contract Address

0xFD98e0fD21785A0b0444E8997e8CC91cf4EB61aB

Client contact

DripX Token Team

Blockchain

Binance Smart chain

Project website

https://www.dripx.win/

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 the **DripX Miners Token Team** to perform an audit of the smart contract.

https://bscscan.com/address/0xFD98e0fD21785A0b0444E8997e8CC91cf4EB61aB

This audit focuses on verifying 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, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Contract details

Token contract details for 07th of March 2024

Contract name	**DRIP MINERS**
Contract address	**0xFD98e0fD21785A0b0444E8997e8CC91cf4EB61aB**
Token supply	**48,733**
Token ticker	**DRIPX.M**
Token holders	**5,141**
Transaction count	**49,122**

Contract code function details

№	Category	Item	Result
1	Coding conventions	ERC20 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	LOW ISSUE
		Overriding variables	PASS
2	Function call audit	Authorization of function call	PASS
		Low level function (call/delegate call) security	PASS
		Returned value security	PASS
		Self destruct function security	PASS
3	Business security & centralisation	Access control of owners	MEDIUM ISSUE
		Business logics	PASS
		Business implementation	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

The 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 their visibility and mutability.

Contract	**Type**	**Bases**		
L	Function Name	Visibility	Mutability	Modifiers
TfiExample	**Implementation**	**Initializable, Ownable Upgradeable, TfiClient**		
L	initialize	Public ❗	🛑	initializer
L	doRequest	Public ❗	🛑	NO ❗
L	doTransferAndRequest	Public ❗	🛑	NO ❗
L	fulfillBytes	Public ❗	🛑	recordChain linkFulfillment
L	changeOracle	Public ❗	🛑	onlyOwner
L	changeJobId	Public ❗	🛑	onlyOwner
L	changeFee	Public ❗	🛑	onlyOwner
L	changeToken	Public ❗	🛑	onlyOwner
L	getToken	Public ❗		NO ❗
L	getChainlinkToken	Public ❗		NO ❗
L	withdrawLink	Public ❗	🛑	onlyOwner
L	typeAndVersion	External ❗		NO ❗
L	renounceOwnership	Public ❗		onlyOwner
TfiClient	**Implementation**			
L	__TfiClient_init	Public ❗	🛑	NO ❗
L	buildChainlinkRequest	Internal 🔒		
L	buildOperatorRequest	Internal 🔒		

└	sendChainlinkRequest	Internal 🔒	🛑	
└	sendChainlinkRequestTo	Internal 🔒	🛑	
└	sendOperatorRequest	Internal 🔒	🛑	
└	sendOperatorRequestTo	Internal 🔒	🛑	
└	_rawRequest	Private 🔐	🛑	
└	cancelChainlinkRequest	Internal 🔒	🛑	
└	getNextRequestCount	Internal 🔒		
└	setChainlinkOracle	Internal 🔒	🛑	
└	setChainlinkToken	Internal 🔒	🛑	
└	setPublicChainlinkToken	Internal 🔒	🛑	
└	chainlinkTokenAddress	Internal 🔒		
└	chainlinkOracleAddress	Internal 🔒		
└	addChainlinkExternalRequest	Internal 🔒	🛑	notPending Request
└	useChainlinkWithENS	Internal 🔒	🛑	
└	updateChainlinkOracleWithENS	Internal 🔒	🛑	
└	validateChainlinkCallback	Internal 🔒	🛑	recordChainlinkFulfillment
Strings	**Library**			
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
Ownable Upgradeable	**Implementation**	**Initializable, Context Upgradeable**		
└	__Ownable_init	Internal 🔒	🛑	only Initializing
└	__Ownable_init_unchained	Internal 🔒	🛑	only Initializing

L	owner	Public ❗		NO ❗
L	_checkOwner	Internal 🔒		
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	
Initializable	**Implementation**			
L	_disableInitializers	Internal 🔒	🛑	
OperatorInterface	**Interface**	**OracleInterface, Chainlink Request Interface**		
L	operatorRequest	External ❗	🛑	NO ❗
L	fulfillOracleRequest2	External ❗	🛑	NO ❗
L	ownerTransferAndCall	External ❗	🛑	NO ❗
L	distributeFunds	External ❗	💵	NO ❗
L	getAuthorizedSenders	External ❗	🛑	NO ❗
L	setAuthorizedSenders	External ❗	🛑	NO ❗
L	getForwarder	External ❗	🛑	NO ❗
ENSInterface	**Interface**			
L	setSubnodeOwner	External ❗	🛑	NO ❗
L	setResolver	External ❗	🛑	NO ❗
L	setOwner	External ❗	🛑	NO ❗
L	setTTL	External ❗	🛑	NO ❗
L	owner	External ❗		NO ❗
L	resolver	External ❗		NO ❗
L	ttl	External ❗		NO ❗

ChainlinkRequest Interface	**Interface**			
└	oracleRequest	External ❗	🛑	NO ❗
└	cancelOracleRequest	External ❗	🛑	NO ❗
Chainlink	**Library**			
└	initialize	Internal 🔒		
└	setBuffer	Internal 🔒		
└	add	Internal 🔒		
└	addBytes	Internal 🔒		
└	addInt	Internal 🔒		
└	addUint	Internal 🔒		
└	addStringArray	Internal 🔒		
LinkToken Interface	**Interface**			
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	balanceOf	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	decreaseApproval	External ❗	🛑	NO ❗
└	increaseApproval	External ❗	🛑	NO ❗
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	totalSupply	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferAndCall	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
ENSResolver	**Implementation**			

L	addr	Public ❗		NO ❗
CBORChainlink	**Library**			
L	encodeFixedNumeric	Private 🔐		
L	encodeIndefiniteLengthType	Private 🔐		
L	encodeUInt	Internal 🔒		
L	encodeInt	Internal 🔒		
L	encodeBytes	Internal 🔒		
L	encodeBigNum	Internal 🔒		
L	encodeSignedBigNum	Internal 🔒		
L	encodeString	Internal 🔒		
L	startArray	Internal 🔒		
L	startMap	Internal 🔒		
L	endSequence	Internal 🔒		
BufferChainlink	**Library**			
L	init	Internal 🔒		
L	fromBytes	Internal 🔒		
L	resize	Private 🔐		
L	max	Private 🔐		
L	truncate	Internal 🔒		
L	write	Internal 🔒		
L	append	Internal 🔒		
L	append	Internal 🔒		
L	writeUint8	Internal 🔒		
L	appendUint8	Internal 🔒		
L	write	Private 🔐		
L	writeBytes20	Internal 🔒		
L	appendBytes20	Internal 🔒		

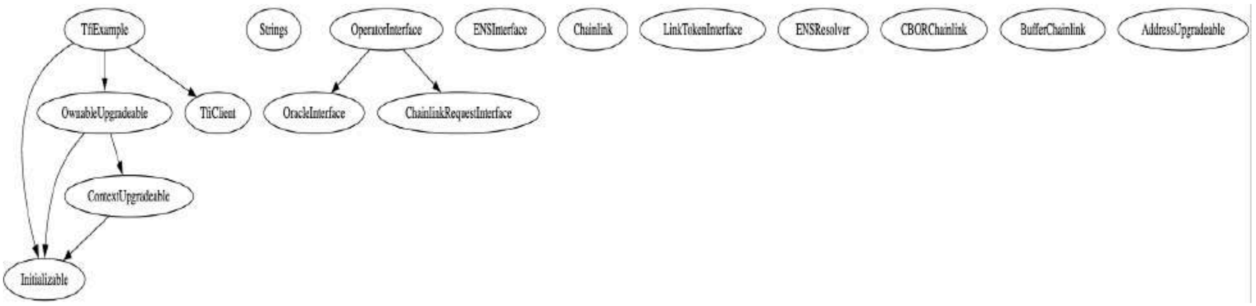
└	appendBytes32	Internal 🔒		
└	writeInt	Private 🔐		
└	appendInt	Internal 🔒		
OracleInterface	**Interface**			
└	fulfillOracleRequest	External ❗	🛑	NO ❗
└	isAuthorizedSender	External ❗		NO ❗
└	withdraw	External ❗	🛑	NO ❗
└	withdrawable	External ❗		NO ❗
Context Upgradeable	**Implementation**	**Initializable**		
└	__Context_init	Internal 🔒	🛑	only Initializing
└	__Context_init_unchained	Internal 🔒	🛑	only Initializing
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address Upgradeable	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	verifyCallResult	Internal 🔒		

CBORChainlink	Library			
L	encodeFixedNumeric	Private 🔐		
L	encodeIndefiniteLengthType	Private 🔐		
L	encodeUInt	Internal 🔒		
L	encodeInt	Internal 🔒		
L	encodeBytes	Internal 🔒		
L	encodeBigNum	Internal 🔒		
L	encodeSignedBigNum	Internal 🔒		
L	encodeString	Internal 🔒		
L	startArray	Internal 🔒		
L	startMap	Internal 🔒		
L	endSequence	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy

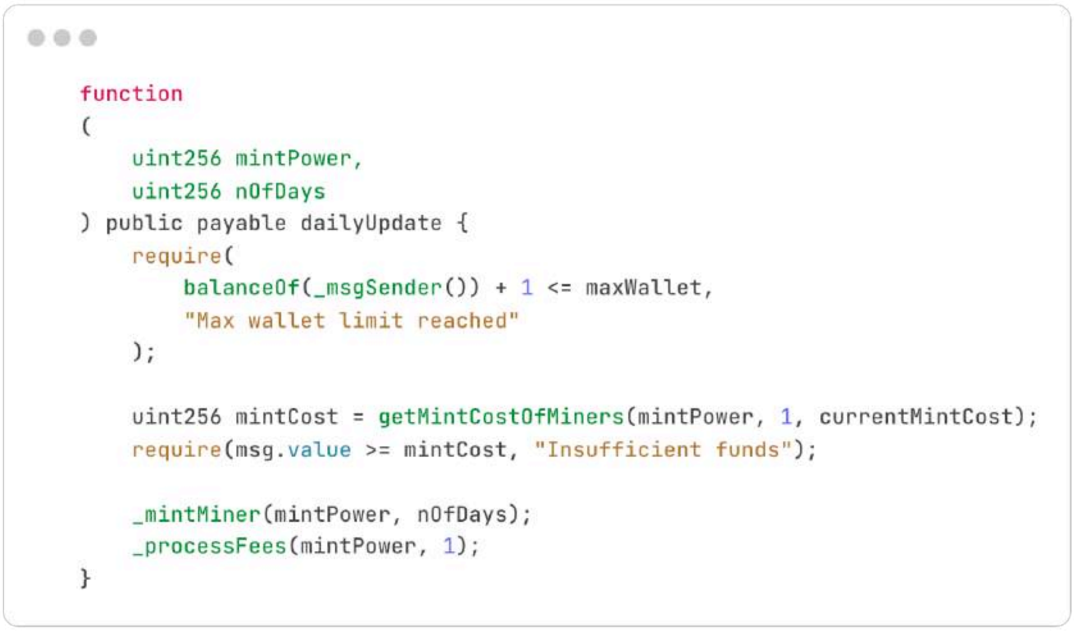


Security issue checking status

❖ High severity issues

The contract is an upgradable contract, the owner can change the functions later

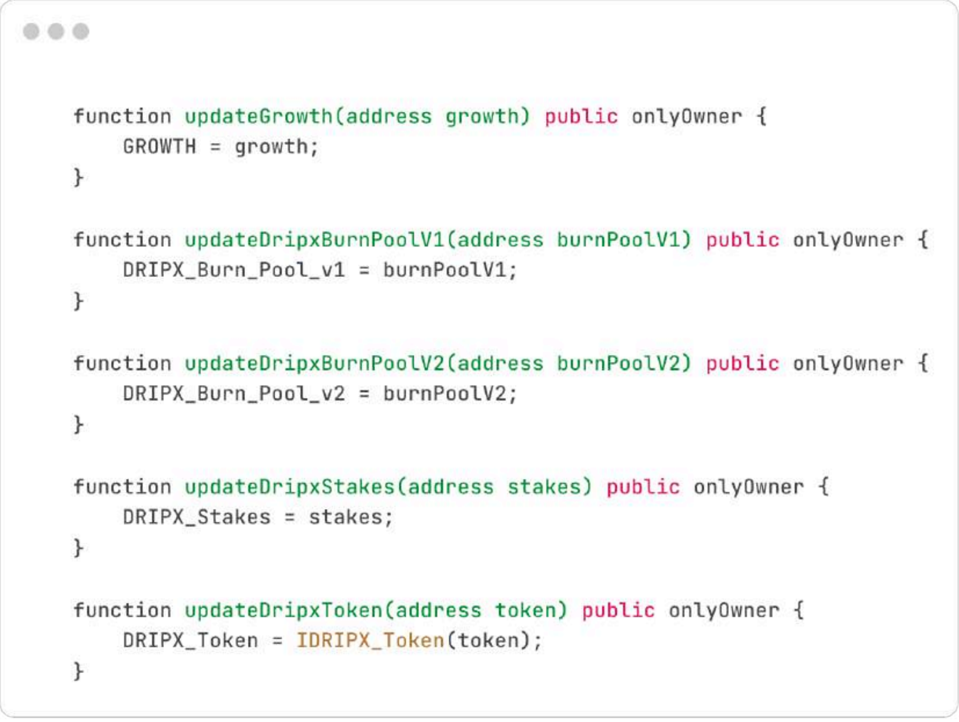
The function on line number 411 lacks a function name; consequently, it will not function properly and will fail to compile. Fixed



```
function
(
    uint256 mintPower,
    uint256 nOfDays
) public payable dailyUpdate {
    require(
        balanceOf(_msgSender()) + 1 <= maxWallet,
        "Max wallet limit reached"
    );

    uint256 mintCost = getMintCostOfMiners(mintPower, 1, currentMintCost);
    require(msg.value >= mintCost, "Insufficient funds");

    _mintMiner(mintPower, nOfDays);
    _processFees(mintPower, 1);
}
```

❖ Medium severity issues

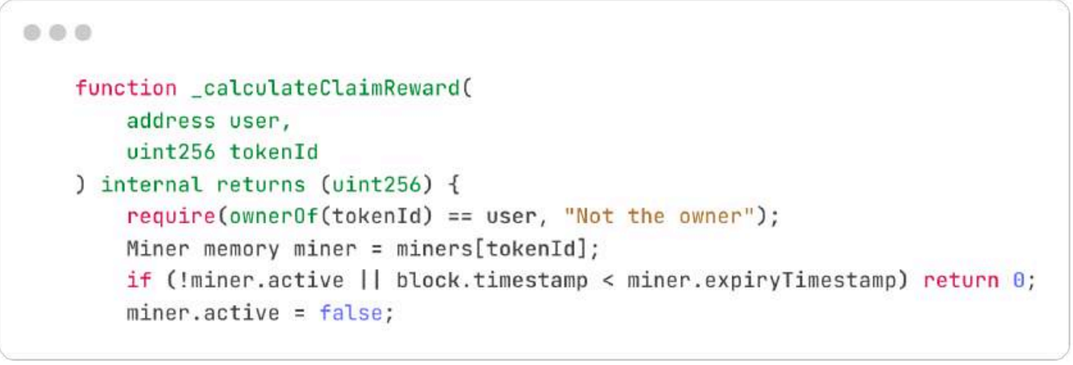
When updating growth, burn pools, staking, and drip tokens, the system does not validate those addresses. If the owner sets them to contracts that cannot receive ETH, the new miner registration process will fail.



```
function updateGrowth(address growth) public onlyOwner {
    GROWTH = growth;
}

function updateDripxBurnPoolV1(address burnPoolV1) public onlyOwner {
    DRIPX_Burn_Pool_v1 = burnPoolV1;
}

function updateDripxBurnPoolV2(address burnPoolV2) public onlyOwner {
    DRIPX_Burn_Pool_v2 = burnPoolV2;
}

function updateDripxStakes(address stakes) public onlyOwner {
    DRIPX_Stakes = stakes;
}

function updateDripxToken(address token) public onlyOwner {
    DRIPX_Token = IDRIPX_Token(token);
}
```

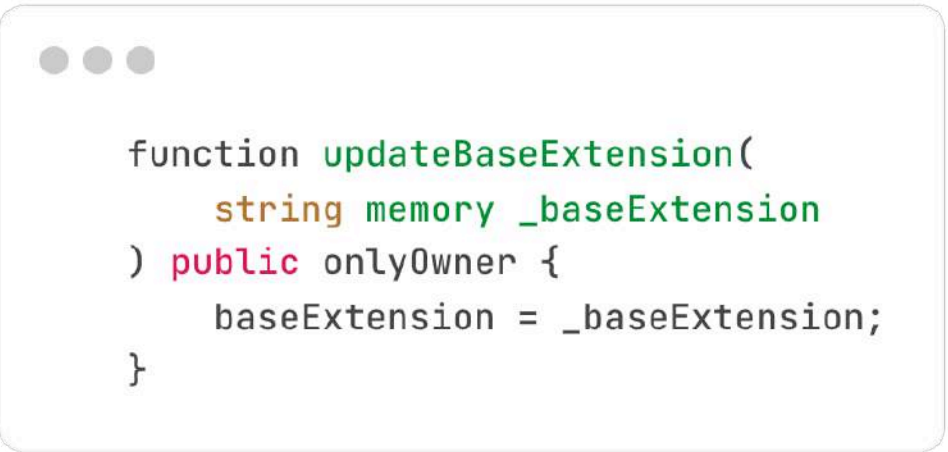
❖ Low severity issues

When sending rewards to miners, the contract simply changes the token status to false. It is advisable to also burn the current NFT for better management.



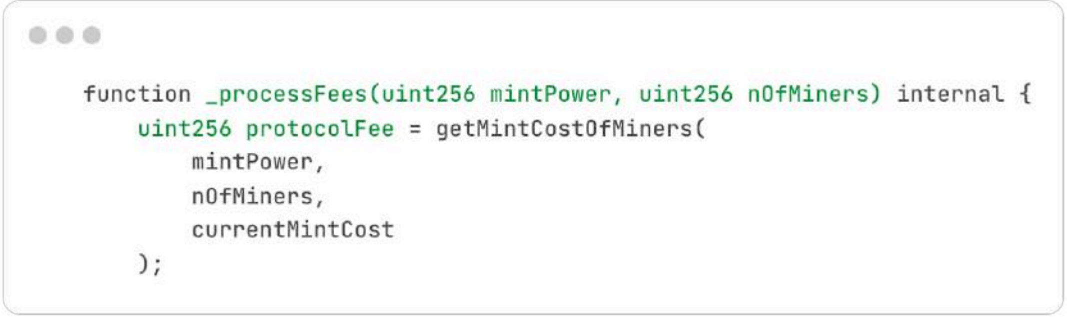
```
function _calculateClaimReward(
    address user,
    uint256 tokenId
) internal returns (uint256) {
    require(ownerOf(tokenId) == user, "Not the owner");
    Miner memory miner = miners[tokenId];
    if (!miner.active || block.timestamp < miner.expiryTimestamp) return 0;
    miner.active = false;
```

The owner can set the base extension to any value. If the owner sets it to the wrong value, the NFT metadata will not be retrieved.



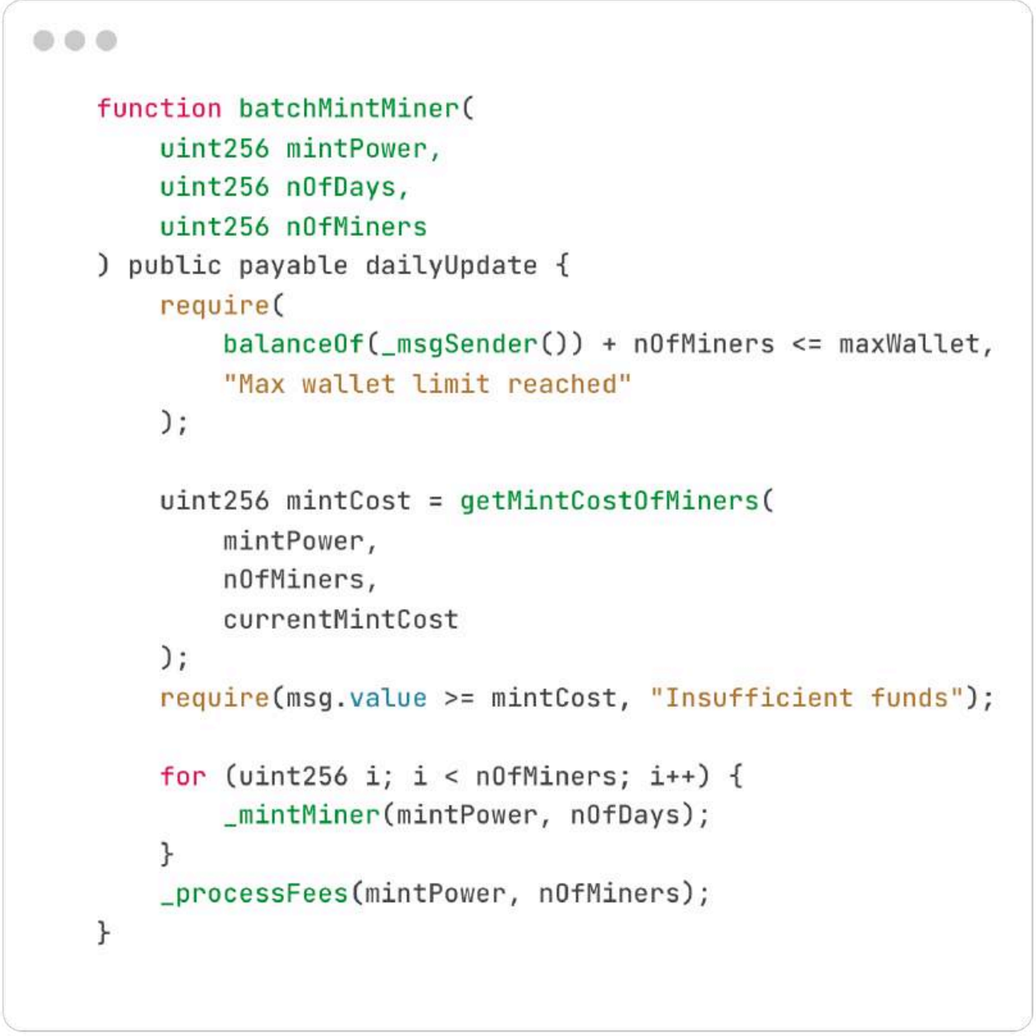
```
function updateBaseExtension(
    string memory _baseExtension
) public onlyOwner {
    baseExtension = _baseExtension;
}
```

The protocol fee is already being calculated and validated in the MintMiner functions; therefore, these statements here are redundant.



```
function _processFees(uint256 mintPower, uint256 nOfMiners) internal {
    uint256 protocolFee = getMintCostOfMiners(
        mintPower,
        nOfMiners,
        currentMintCost
    );
```

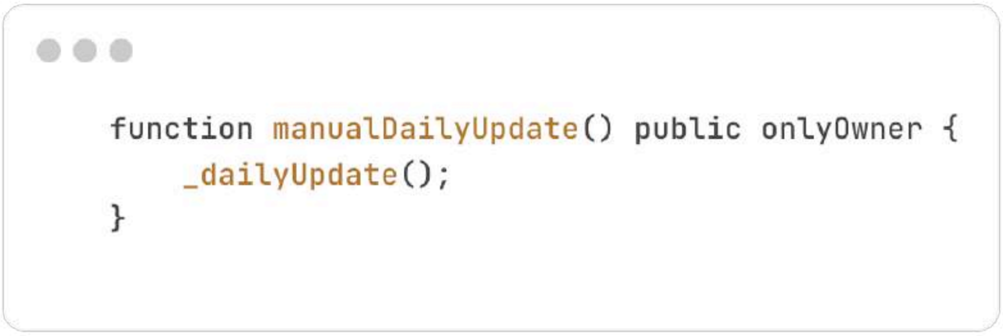
In the batchMintMiner function, there is no maximum limit set for one-time usage. If users employ a large number of miners at once, this loop may fail due to reaching the maximum gas limit.



```
function batchMintMiner(
    uint256 mintPower,
    uint256 nOfDays,
    uint256 nOfMiners
) public payable dailyUpdate {
    require(
        balanceOf(_msgSender()) + nOfMiners <= maxWallet,
        "Max wallet limit reached"
    );

    uint256 mintCost = getMintCostOfMiners(
        mintPower,
        nOfMiners,
        currentMintCost
    );
    require(msg.value >= mintCost, "Insufficient funds");

    for (uint256 i; i < nOfMiners; i++) {
        _mintMiner(mintPower, nOfDays);
    }
    _processFees(mintPower, nOfMiners);
}
```

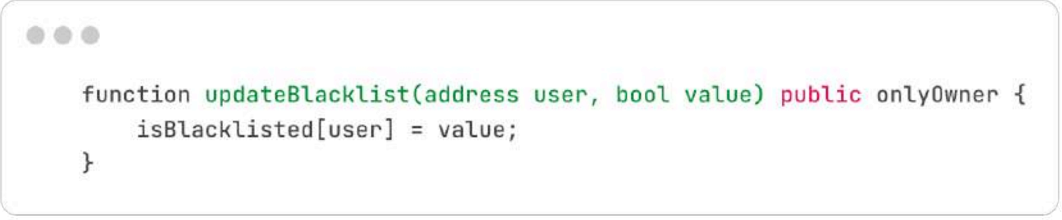
Owner privileges

- Owner can manually update the contract stats



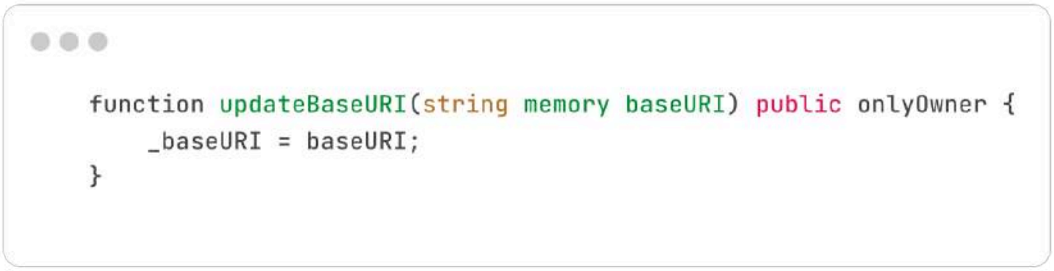
```
function manualDailyUpdate() public onlyOwner {
    _dailyUpdate();
}
```

- Owner can blacklist/whitelist users from contract



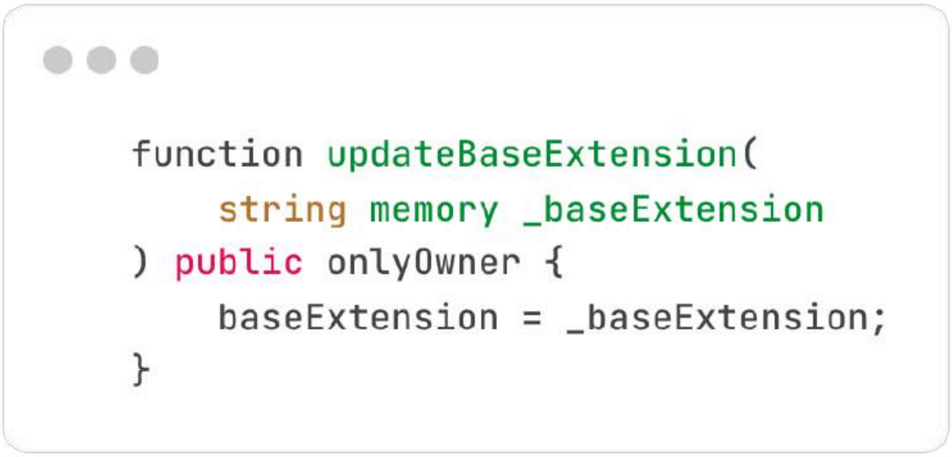
```
function updateBlacklist(address user, bool value) public onlyOwner {
    isBlacklisted[user] = value;
}
```

- Owner can update base URI



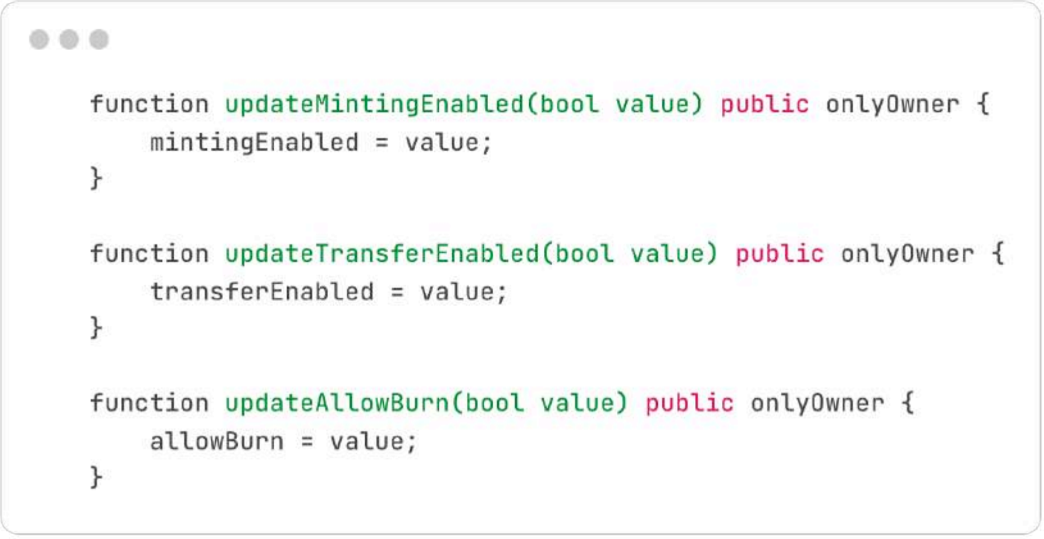
```
function updateBaseURI(string memory baseURI) public onlyOwner {
    _baseURI = baseURI;
}
```

- ❖ Owner can update base extension



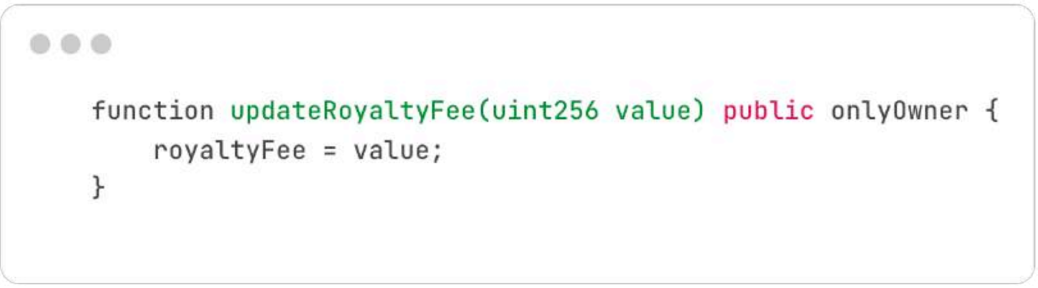
```
function updateBaseExtension(
    string memory _baseExtension
) public onlyOwner {
    baseExtension = _baseExtension;
}
```

- ❖ Owner can enable/disable minting,transfers and burns



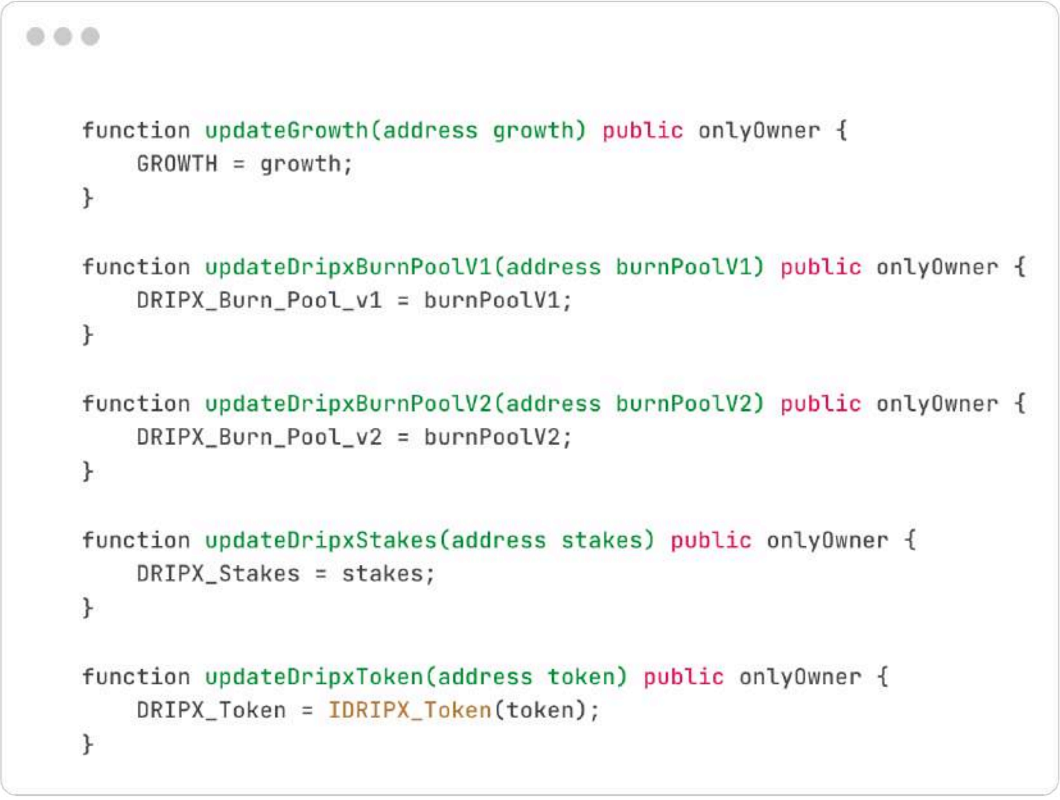
```
function updateMintingEnabled(bool value) public onlyOwner {
    mintingEnabled = value;
}

function updateTransferEnabled(bool value) public onlyOwner {
    transferEnabled = value;
}

function updateAllowBurn(bool value) public onlyOwner {
    allowBurn = value;
}
```

- ❖ Owner can update royalty fees



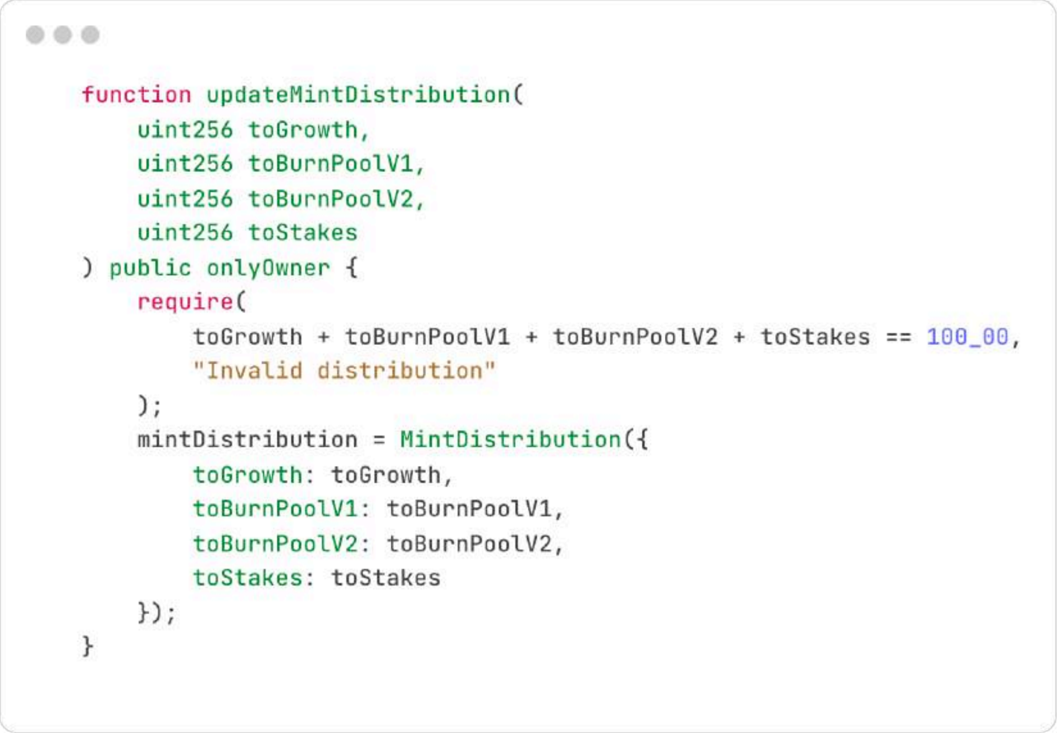
```
function updateRoyaltyFee(uint256 value) public onlyOwner {
    royaltyFee = value;
}
```

- ❖ Owner can change growth,burn pools,stake and drip token contracts



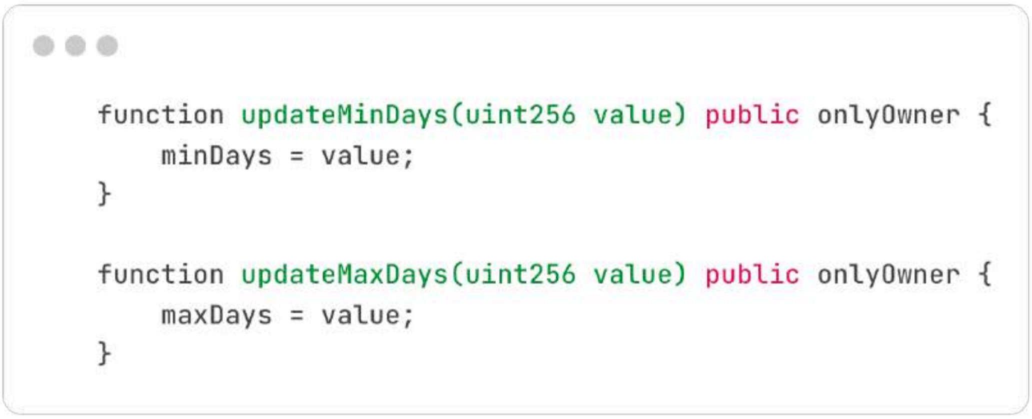
```
function updateGrowth(address growth) public onlyOwner {
    GROWTH = growth;
}

function updateDripxBurnPoolV1(address burnPoolV1) public onlyOwner {
    DRIPX_Burn_Pool_v1 = burnPoolV1;
}

function updateDripxBurnPoolV2(address burnPoolV2) public onlyOwner {
    DRIPX_Burn_Pool_v2 = burnPoolV2;
}

function updateDripxStakes(address stakes) public onlyOwner {
    DRIPX_Stakes = stakes;
}

function updateDripxToken(address token) public onlyOwner {
    DRIPX_Token = IDRIPX_Token(token);
}
```

- Owner can change fee distribution criteria



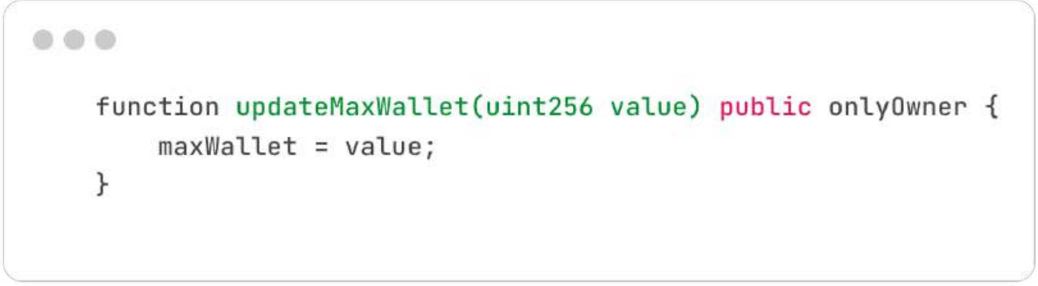
```
function updateMintDistribution(
    uint256 toGrowth,
    uint256 toBurnPoolV1,
    uint256 toBurnPoolV2,
    uint256 toStakes
) public onlyOwner {
    require(
        toGrowth + toBurnPoolV1 + toBurnPoolV2 + toStakes == 100_00,
        "Invalid distribution"
    );
    mintDistribution = MintDistribution({
        toGrowth: toGrowth,
        toBurnPoolV1: toBurnPoolV1,
        toBurnPoolV2: toBurnPoolV2,
        toStakes: toStakes
    });
}
```

- Owner can update minimum and maximum mine days



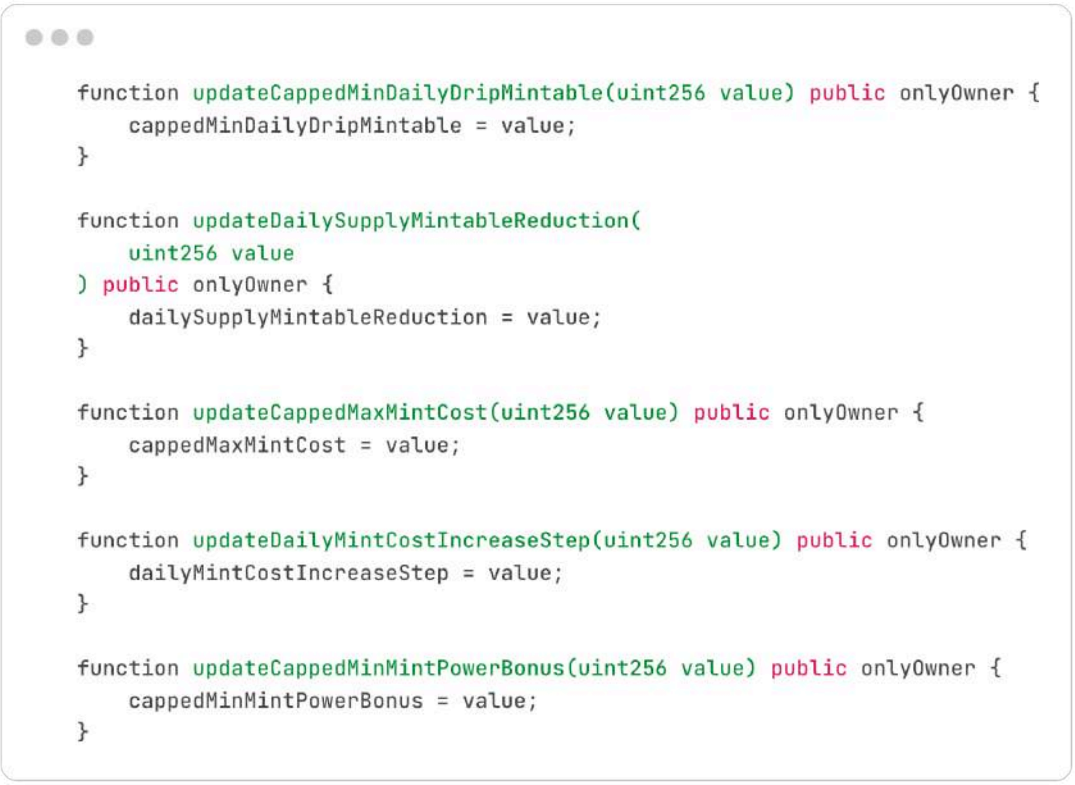
```
function updateMinDays(uint256 value) public onlyOwner {
    minDays = value;
}

function updateMaxDays(uint256 value) public onlyOwner {
    maxDays = value;
}
```

- ❖ Owner can update max miners per wallet



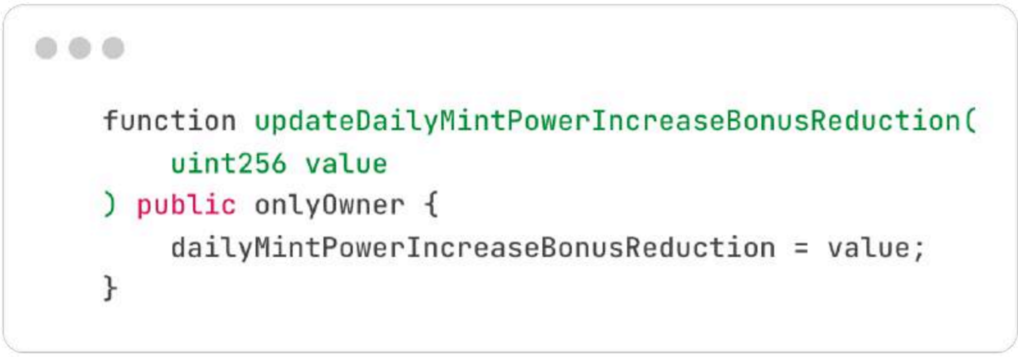
```
function updateMaxWallet(uint256 value) public onlyOwner {
    maxWallet = value;
}
```

- ❖ Owner can change min daily minting token,daily supply reduction,max mint cost,and min mint power bonus



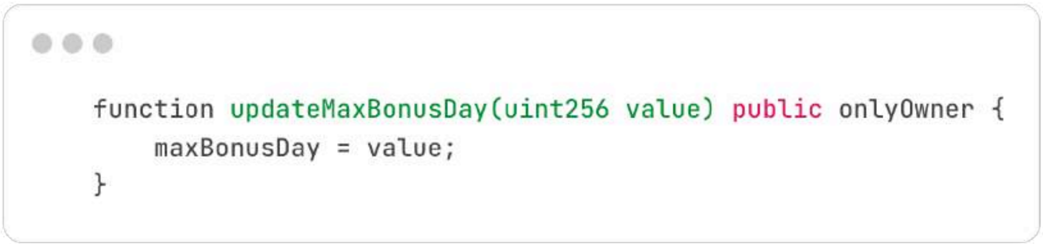
```
function updateCappedMinDailyDripMintable(uint256 value) public onlyOwner {
    cappedMinDailyDripMintable = value;
}

function updateDailySupplyMintableReduction(
    uint256 value
) public onlyOwner {
    dailySupplyMintableReduction = value;
}

function updateCappedMaxMintCost(uint256 value) public onlyOwner {
    cappedMaxMintCost = value;
}

function updateDailyMintCostIncreaseStep(uint256 value) public onlyOwner {
    dailyMintCostIncreaseStep = value;
}

function updateCappedMinMintPowerBonus(uint256 value) public onlyOwner {
    cappedMinMintPowerBonus = value;
}
```

❖ Owner can change mint power increase bonus



```
function updateDailyMintPowerIncreaseBonusReduction(
    uint256 value
) public onlyOwner {
    dailyMintPowerIncreaseBonusReduction = value;
}
```

❖ Owner can change max bonus days



```
function updateMaxBonusDay(uint256 value) public onlyOwner {
    maxBonusDay = value;
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testing, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status:	PASS
Smart contract security Status:	MEDIUM & LOW ISSUES
Number of risk issues:	06
Solidity code functional issue level:	PASS
Number of owner privileges:	13
Centralization risk correlated to the active owner:	HIGH
Smart contract active ownership:	ACTIVE