DripX Buy & Burn Contract

RugfreeCoins Verified on March 07th, 2024

Overview

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

Contents

Audit details

Audited project

DripX Buy & Burn Contract

Contract Address

V1: 0x19C72FcDaB869518AFcbcf0d02c6516d7f1C4c97
V2: 0xc56E00A01F6eE5c4eD588B8673977a551Ed0dB17

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 Team** to perform an audit of the smart contract.

V1: https://bscscan.com/address/0x19C72FcDaB869518AFcbcf0d02c6516d7f1C4c97
V2: https://bscscan.com/address/0xc56E00A01F6eE5c4eD588B8673977a551Ed0dB17

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 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	PASS
		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
		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		
└	Function Name	Visibility	Mutability	Modifiers
AdminUpgradeability Proxy	**Implementation**	**Transparent Upgradeable Proxy**		
└		Public ❗	💵	Transparent Upgradeable Proxy
Transparent UpgradeableProxy	**Implementation**	**ERC1967 Proxy**		
└		Public ❗	💵	ERC1967Proxy
└	admin	External ❗	🛑	ifAdmin
└	implementation	External ❗	🛑	ifAdmin
└	changeAdmin	External ❗	🛑	ifAdmin
└	upgradeTo	External ❗	🛑	ifAdmin
└	upgradeToAndCall	External ❗	💵	ifAdmin
└	_admin	Internal 🔒		
└	_beforeFallback	Internal 🔒	🛑	
BeaconProxy	**Implementation**	**Proxy, ERC1967 Upgrade**		
└		Public ❗	💵	NO ❗
└	_beacon	Internal 🔒		
└	_implementation	Internal 🔒		
└	_setBeacon	Internal 🔒	🛑	

UpgradeableBeacon	**Implementation**	**IBeacon, Ownable**		
L		Public ❗	🛑	NO ❗
L	implementation	Public ❗		NO ❗
L	upgradeTo	Public ❗	🛑	onlyOwner
L	_setImplementation	Private 🔐	🛑	
ERC1967Proxy	**Implementation**	**Proxy, ERC1967Upgrade**		
L		Public ❗	💵	NO ❗
L	_implementation	Internal 🔒		
ProxyAdmin	**Implementation**	**Ownable**		
L	getProxyImplementation	Public ❗		NO ❗
L	getProxyAdmin	Public ❗		NO ❗
L	changeProxyAdmin	Public ❗	🛑	onlyOwner
L	upgrade	Public ❗	🛑	onlyOwner
L	upgradeAndCall	Public ❗	💵	onlyOwner
IBeacon	**Interface**			
L	implementation	External ❗		NO ❗
Proxy	**Implementation**			
L	_delegate	Internal 🔒	🛑	
L	_implementation	Internal 🔒		
L	_fallback	Internal 🔒	🛑	
L		External ❗	💵	NO ❗
L		External ❗	💵	NO ❗
L	_beforeFallback	Internal 🔒	🛑	

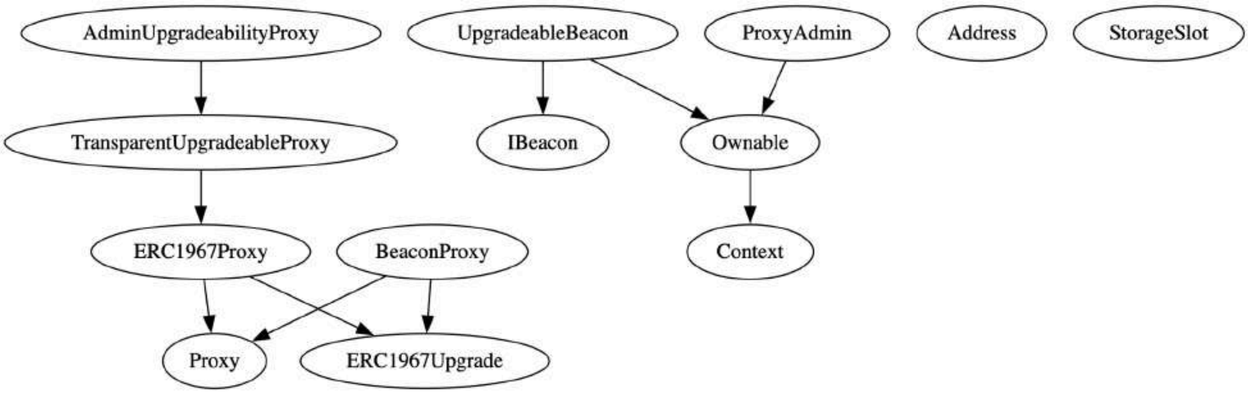
ERC1967Upgrade	**Implementation**			
└	_getImplementation	Internal 🔒		
└	_setImplementation	Private 🔐	🛑	
└	_upgradeTo	Internal 🔒	🛑	
└	_upgradeToAndCall	Internal 🔒	🛑	
└	_upgradeToAndCallSecure	Internal 🔒	🛑	
└	_upgradeBeaconToAndCall	Internal 🔒	🛑	
└	_getAdmin	Internal 🔒		
└	_setAdmin	Private 🔐	🛑	
└	_changeAdmin	Internal 🔒	🛑	
└	_getBeacon	Internal 🔒		
└	_setBeacon	Private 🔐	🛑	
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	_verifyCallResult	Private 🔐		
StorageSlot	**Library**			
└	getAddressSlot	Internal 🔒		

L	getBooleanSlot	Internal 🔒		
L	getBytes32Slot	Internal 🔒		
L	getUint256Slot	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



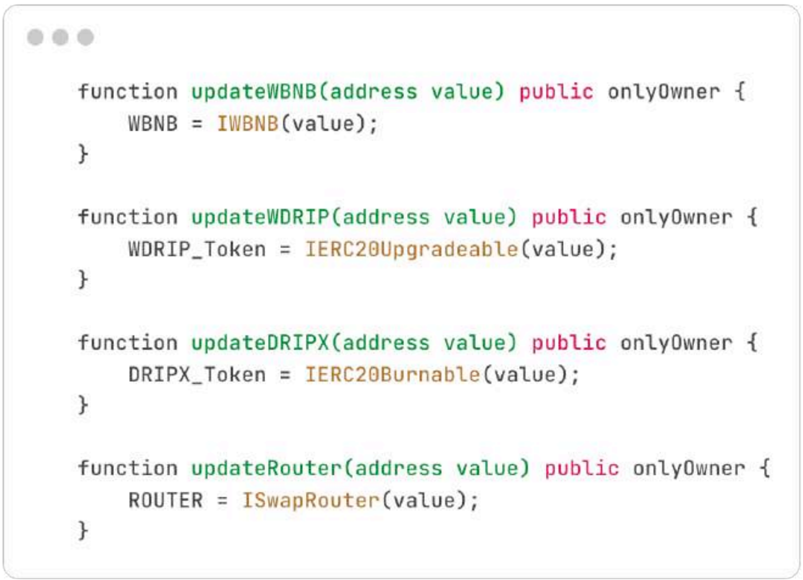
Security issue checking status

❖ High severity issues

No high severity issues

❖ Medium severity issues

The owner can change the WBNB, token, and router addresses. If the owner sets them to the wrong addresses, the contract will fail.



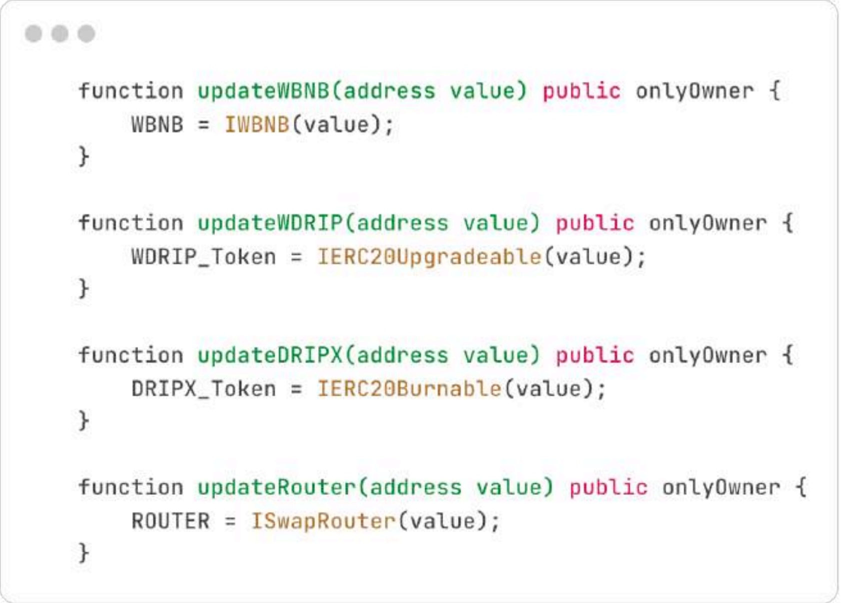
```
function updateWBNB(address value) public onlyOwner {
    WBNB = IWBNB(value);
}

function updateWDRIP(address value) public onlyOwner {
    WDRIP_Token = IERC20Upgradeable(value);
}

function updateDRIPX(address value) public onlyOwner {
    DRIPX_Token = IERC20Burnable(value);
}

function updateRouter(address value) public onlyOwner {
    ROUTER = ISwapRouter(value);
}
```

❖ Low severity issues

No low severity issues

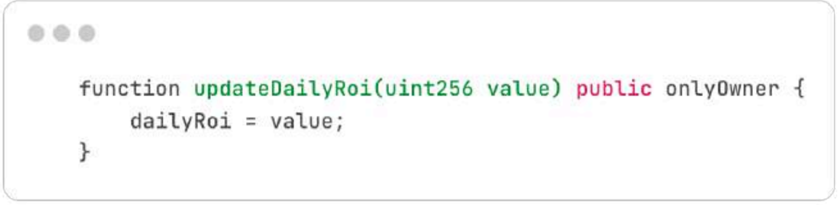
Owner privileges

- ❖ Owner can change WBNB, Drip tokens and router address



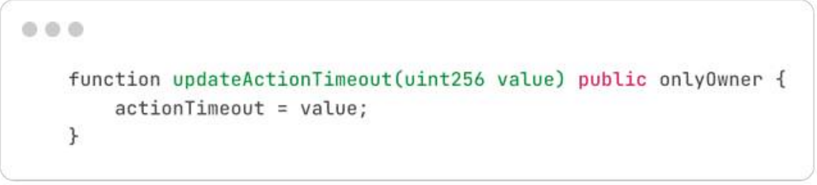
```
function updateWBNB(address value) public onlyOwner {
    WBNB = IWBNB(value);
}

function updateWDRIP(address value) public onlyOwner {
    WDRIP_Token = IERC20Upgradeable(value);
}

function updateDRIPX(address value) public onlyOwner {
    DRIPX_Token = IERC20Burnable(value);
}

function updateRouter(address value) public onlyOwner {
    ROUTER = ISwapRouter(value);
}
```

- ❖ Owner can change the daily roi



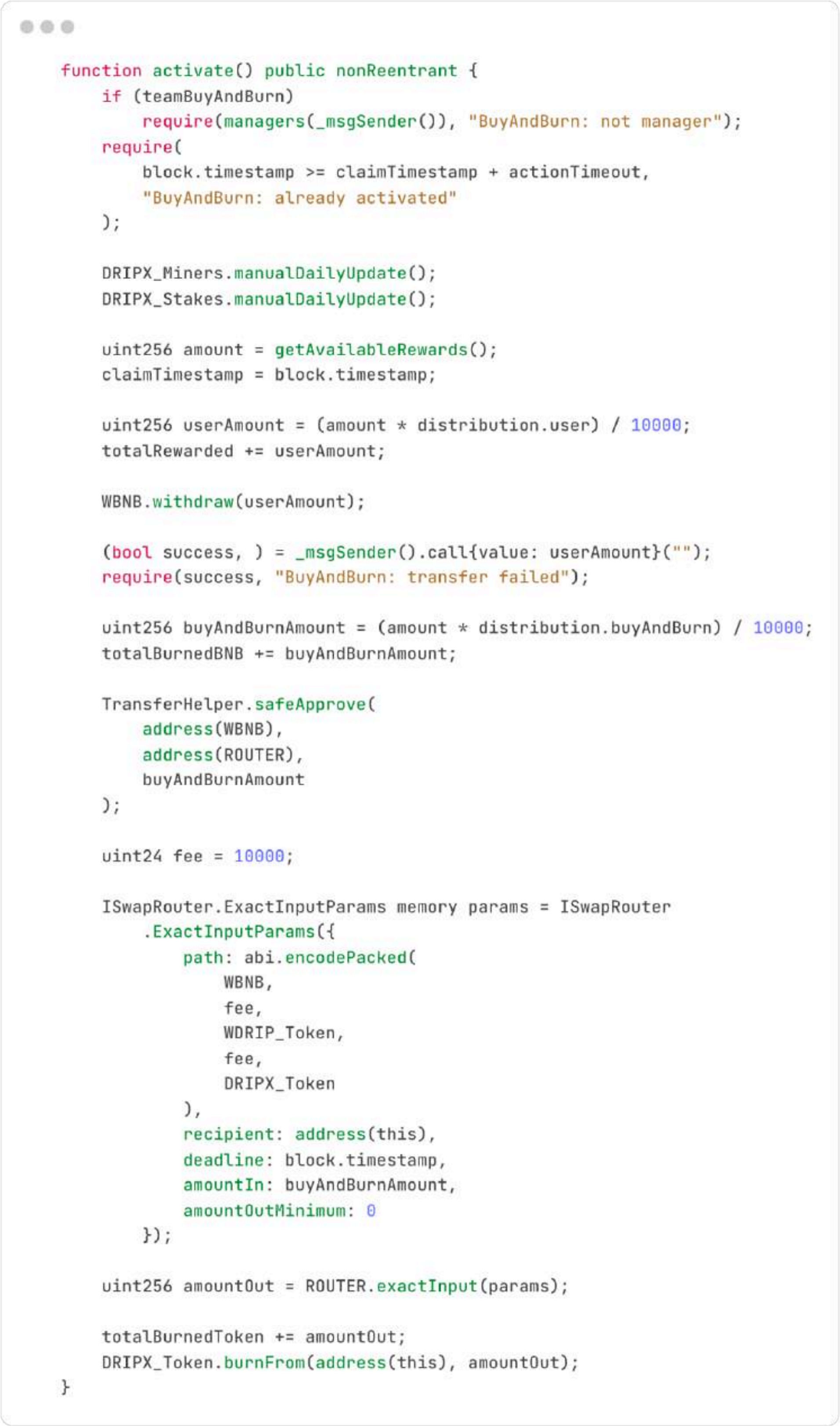
```
function updateDailyRoi(uint256 value) public onlyOwner {
    dailyRoi = value;
}
```

- ❖ Owner can change the activate timeout



```
function updateActionTimeout(uint256 value) public onlyOwner {
    actionTimeout = value;
}
```

- Managers can activate the buy and burn pool



```
function activate() public nonReentrant {
    if (teamBuyAndBurn)
        require(managers(_msgSender()), "BuyAndBurn: not manager");
    require(
        block.timestamp >= claimTimestamp + actionTimeout,
        "BuyAndBurn: already activated"
    );

    DRIPX_Miners.manualDailyUpdate();
    DRIPX_Stakes.manualDailyUpdate();

    uint256 amount = getAvailableRewards();
    claimTimestamp = block.timestamp;

    uint256 userAmount = (amount * distribution.user) / 10000;
    totalRewarded += userAmount;

    WBNB.withdraw(userAmount);

    (bool success, ) = _msgSender().call{value: userAmount}("");
    require(success, "BuyAndBurn: transfer failed");

    uint256 buyAndBurnAmount = (amount * distribution.buyAndBurn) / 10000;
    totalBurnedBNB += buyAndBurnAmount;

    TransferHelper.safeApprove(
        address(WBNB),
        address(ROUTER),
        buyAndBurnAmount
    );

    uint24 fee = 10000;

    ISwapRouter.ExactInputParams memory params = ISwapRouter
        .ExactInputParams({
            path: abi.encodePacked(
                WBNB,
                fee,
                WDRIP_Token,
                fee,
                DRIPX_Token
            ),
            recipient: address(this),
            deadline: block.timestamp,
            amountIn: buyAndBurnAmount,
            amountOutMinimum: 0
        });

    uint256 amountOut = ROUTER.exactInput(params);

    totalBurnedToken += amountOut;
    DRIPX_Token.burnFrom(address(this), amountOut);
}
```

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 ISSUE
Number of risk issues:	01
Solidity code functional issue level:	PASS
Number of owner privileges:	04
Centralization risk correlated to the active owner:	HIGH
Smart contract active ownership:	ACTIVE