Cyberscope

Audit Report

XDOGECOIN

September 2023

Network	BSC
Address	0x5816F112fFd3378e56c2775B23061425B9ac5119
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	MVN	Misleading Variables Naming	Unresolved
●	MRM	Missing Revert Messages	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	CR	Code Repetition	Unresolved
●	RTLF	Redundant Time Lock-Exempt Functionality	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	RCF	Redundant Cooldown Functionality	Unresolved
●	RKBF	Redundant Kill Block Functionality	Unresolved
●	RMWF	Redundant Max Wallet Functionality	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	XDOGECOIN
Compiler Version	v0.7.4+commit.3f05b770
Optimization	200 runs
Explorer	https://bscscan.com/address/0x5816f112ffd3378e56c2775b23061425b9ac5119
Address	0x5816f112ffd3378e56c2775b23061425b9ac5119
Network	BSC
Symbol	XDOGE
Decimals	9
Total Supply	10,000,000,000

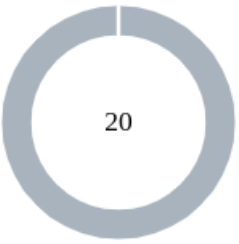
Audit Updates

Initial Audit	27 Sep 2023 https://github.com/cyberscope-io/audits/blob/main/2-xdoge/v1/audit.pdf
Corrected Phase 2	03 Oct 2023

Source Files

Filename	SHA256
XDOGECOIN.sol	2fc0e68dcccfd19f51f45c61cb8bf5e86f6bf45683500c82737c2423697121fe

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	20

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	20	0	0	0

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L640
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

The variable `burnFeeReceiver` is used to demonstrate the dead address. But it could be mutated. Thus, the variable name is misleading.

```
function setFeeReceivers(address _autoLiquidityReceiver, address
_marketingFeeReceiver, address _devfeeReceiver, address
_burnFeeReceiver ) external onlyOwner {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
    devfeeReceiver = _devfeeReceiver;
    burnFeeReceiver = _burnFeeReceiver;
}
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

MRM - Missing Revert Messages

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L176,182,604,658
Status	Unresolved

Description

The contract is missing error messages. These missing error messages are making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(holder != address(this) && holder != pair);
require(gas < 750000);
...
```

Recommendation

The team is advised to carefully review the source code in order to address these issues. To accelerate the debugging process and mitigate these issues, the team should use more specific and descriptive error messages.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L613,617,621,636
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

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

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

...
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L613,617,621,636,648,657
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes.

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

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

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

...
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

CR - Code Repetition

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L677,703
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
function multiTransfer(address from, address[] calldata addresses,
uint256[] calldata tokens) external onlyOwner {
    ...
    require(balanceOf(from) >= SCCC, "Not enough tokens in wallet");

    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

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the

contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

RTLF - Redundant Time Lock-Exempt Functionality

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L330,621
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract does not utilize the time lock-exempt functionality in the contract implementation. Hence, it is redundant.

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

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L577
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
uint256 amountBNBReflection =
amountBNB.mul(reflectionFee).div(totalBNBFee);
uint256 amountBNBMarketing =
amountBNB.mul(marketingFee).div(totalBNBFee);
uint256 amountBNBDev = amountBNB.mul(devfee).div(totalBNBFee);
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

RCF - Redundant Cooldown Functionality

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L364,549
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract does not utilize the cooldown functionality in the contract implementation. Hence, it is redundant.

```
bool public buyCooldownEnabled = true;
uint8 public cooldownTimerInterval = 0;
mapping (address => uint) private cooldownTimer;

function cooldownEnabled(bool _status, uint8 _interval) public
onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

RKBF - Redundant Kill Block Functionality

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L355,450
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract does not utilize the killblock functionality in the contract. Hence, it is redundant.

```
uint public killblock = 2;

function setKillBlock(uint num) public onlyOwner {
    killblock = num;
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

RMWF - Redundant Max Wallet Functionality

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L322
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract does not utilize max wallet functionality in the contract implementation. Hence, it is redundant.

```
uint256 public _maxWalletToken = _totalSupply;
function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000)
external onlyOwner() {
    _maxWalletToken = (_totalSupply * maxWallPercent_base1000 ) / 1000;
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L373
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
constructor () Auth(msg.sender) {
        router =
IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
        pair = IDEXFactory(router.factory()).createPair(WBNB,
address(this));
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L374,375,378
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
router
pair
distributor
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L153,154,167,311,312,313,319,358,359
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
IBEP20 RWRD = IBEP20(0x55d398326f99059fF775485246999027B3197955)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
uint256 _totalSupply = 1 * 10**10 * 10**_decimals
uint256 swapAt = 4 * (10 ** 9)
uint256 swapDelay = 7
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L89,145,153,154,192,311,312,313,315,316,317,319,321,322,324,325,439,442,532,549,625,636,643,648,653,703
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
address _token
IBEP20 RWRD = IBEP20(0x55d398326f99059fF775485246999027B3197955)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
string constant _name = "XDOGECOIN"
string constant _symbol = "XDOGE"
uint8 constant _decimals = 9
uint256 _totalSupply = 1 * 10**10 * 10**_decimals
uint256 public _maxTxAmount = _totalSupply
uint256 public _maxWalletToken = _totalSupply

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L358,359,366
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
uint256 swapAt = 4 * (10 ** 9)
uint256 swapDelay = 7
mapping (address => uint) private cooldownTimer
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L193,443,447,533,626,645,649
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
minPeriod = _minPeriod
_maxTxAmount = (_totalSupply * maxTXPercentage_base1000 ) / 1000
_maxTxAmount = amount
sellMultiplier = Multiplier
liquidityFee = _liquidityFee
swapThreshold = _amount
targetLiquidity = _target
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L509,511
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 feeAmount =
amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100)
uint256 burnTokens = feeAmount.mul(burnFee).div(totalFee)
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L76,637,638,639,640
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
owner = adr
autoLiquidityReceiver = _autoLiquidityReceiver
marketingFeeReceiver = _marketingFeeReceiver
devfeeReceiver = _devfeeReceiver
burnFeeReceiver = _burnFeeReceiver
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.7.4;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	XDOGECOIN.sol#L270
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
RWRD.transfer(shareholder, amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-

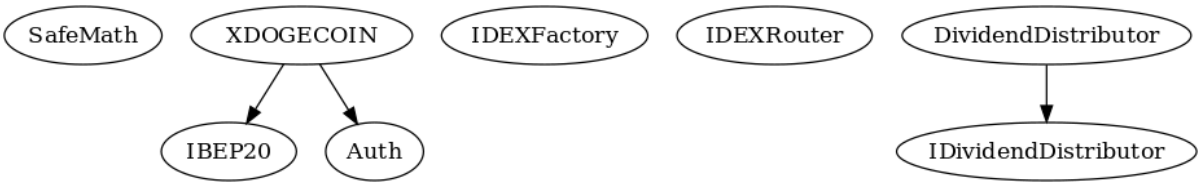
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	isOwner	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-

	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
XDOGECOIN	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-

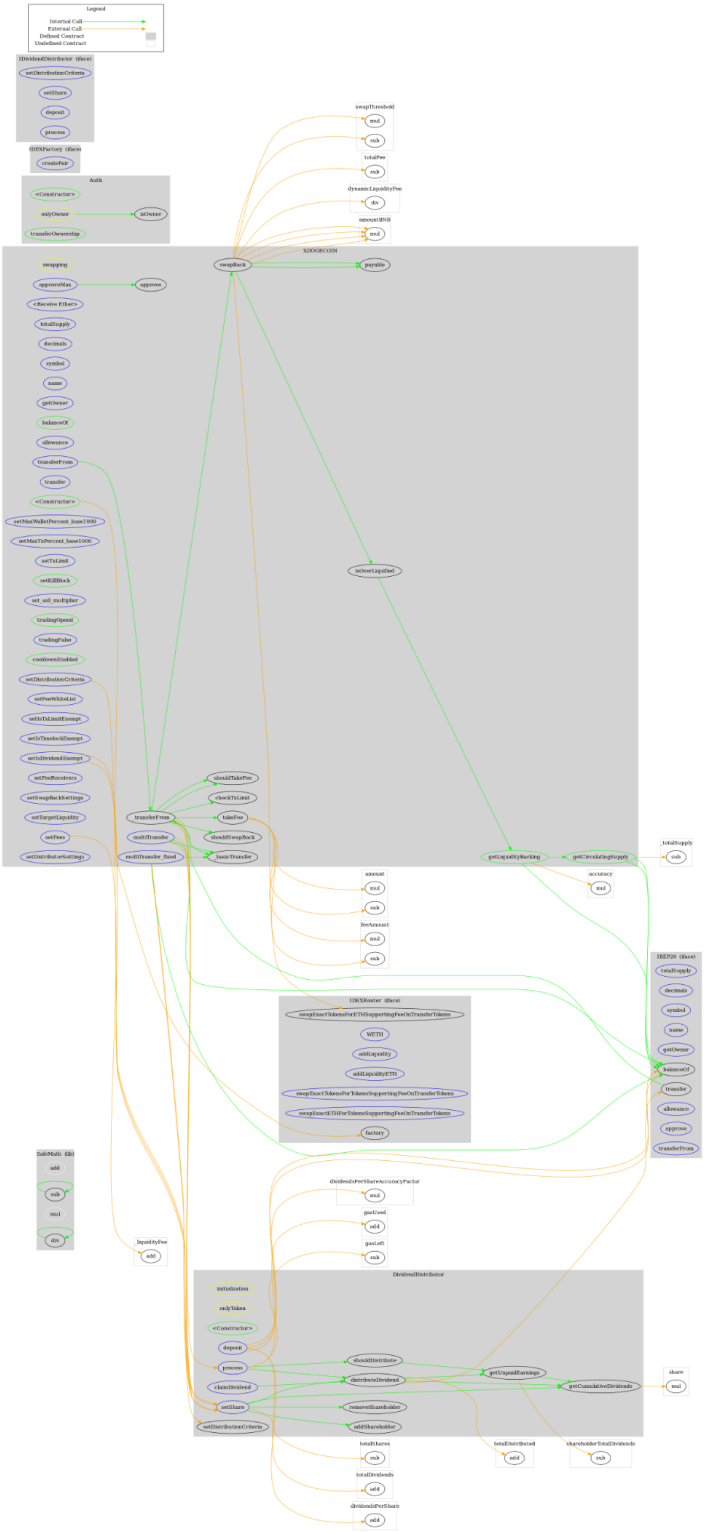
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent_base1000	External	✓	onlyOwner
	setMaxTxPercent_base1000	External	✓	onlyOwner
	setTxLimit	External	✓	onlyOwner
	setKillBlock	Public	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	set_sell_multiplier	External	✓	onlyOwner
	tradingOpend	Public	✓	onlyOwner
	tradingFalse	External	✓	onlyOwner
	cooldownEnabled	Public	✓	onlyOwner
	swapBack	Internal	✓	swapping

	setIsDividendExempt	External	✓	onlyOwner
	setFeeWhiteList	External	✓	onlyOwner
	setIsTxLimitExempt	External	✓	onlyOwner
	setIsTimelockExempt	External	✓	onlyOwner
	setFees	External	✓	onlyOwner
	setFeeReceivers	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
	setTargetLiquidity	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	multiTransfer	External	✓	onlyOwner
	multiTransfer_fixed	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDividendDistributor (iface)
setDistributionCriteria
setShare
deposit
process
IDEXFactory (iface)
createPair
Auth
<Constructor>
onlyOwner
isOwner
transferOwnership
XDOGECOIN
swapping
approveMax
approve
<Receive Ether>
totalSupply
decimals
symbol
name
getOwner
balanceOf
allowance
transferFrom
transfer
<Constructor>
setMaxWalletPercent_base1000
setMaxTxPercent_base1000
setTxLimit
setKillBlock
set_sell_multiplier
tradingOpened
tradingFalse
cooldownEnabled
setDistributionCriteria
setFeeWhiteList
setIsTxLimitExempt
setIsTimelockExempt
setIsDividendExempt
setFeeReceivers
setSwapBackSettings
setTargetLiquidity
setFees
setDistributorSettings
swapThreshold
mul
sub
totalFee
add
dynamicLiquidityFee
div
amountBNB
swapBack
payable
isOverLiquified
shouldTakeFee
checkTxLimit
takeFee
transferFrom
multiTransfer
shouldSwapBack
multiTransfer_fixed
basicTransfer
getLiquidityBacking
getCirculatingSupply
accuracy
amount
feeAmount
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div
liquidityFee
dividendsPerShareAccuracyFactor
gasUsed
gasLeft
DividendDistributor
initialization
onlyToken
<Constructor>
deposit
process
claimDividend
setShare
setDistributionCriteria
shouldDistribute
distributeDividend
removeShareholder
addShareholder
getUnpaidEarnings
getCumulativeDividends
share
totalShares
totalDividends
dividendsPerShare
totalDistributed
shareholderTotalDividends

Summary

XDOGECOIN contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. XDOGECOIN is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues.

The contract's ownership has been renounced. The information regarding the transaction can be accessed through the following link:

https://bscscan.com/tx/0x51ae7a27b2c301f1793f5ee673041507ed1c8c18f2ffa7bf48542a612ce0a102

The fees are locked at 1% on buy, sell and transfer transactions.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io