



Smart Contract Audit

FOR
XDOGE

DATED : 25 July 23'



AUDIT SUMMARY

Project name – XDOGE

Date: 25 July, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	1	0	0	0

USED TOOLS

Tools:

1- Manual Review:

A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :

The code has undergone static analysis using Slither.

Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

<https://testnet.bscscan.com/address/0xd94fc50d37b32c54df0a1c4a34448a92377b5c05>



Token Information

Token Name : XDOGE

Token Symbol: XDOGE

Decimals: 8

Token Supply: 420,000,000,000,000

Token Address:

0xd2b274CfBf9534f56b59aD0FB7e645e0354f4941

Checksum:

050f6e58d9f2fc3bdf49e48c8a8165aa502e7f51

Owner:

0x00

(at time of writing the audit)

Deployer:

0xdC447192E935202FBF4A8Aff3FF95649b4Cd2dA4



TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of the tokens

- modifying fees

- enabling trades (enabled)



AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
 - Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
 - Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
 - Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
 - Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.
-



VULNERABILITY CHECKLIST

- | | |
|--|---|
|  Return values of low-level calls |  Gasless Send |
|  Private modifier |  Using block.timestamp |
|  Multiple Sends |  Re-entrancy |
|  Using Suicide |  Tautology or contradiction |
|  Gas Limitand Loops |  Timestamp Dependence |
|  Address hardcoded |  Revert/require functions |
|  Exception Disorder |  Use of tx.origin |
|  Using inline assembly |  Integer overflow/underflow |
|  Divide before multiply |  Dangerous strict equalities |
|  Missing Zero Address Validation |  Using SHA3 |
|  Compiler version not fixed |  Using throw |
-



CLASSIFICATION OF RISK

Severity

Description

◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization / Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

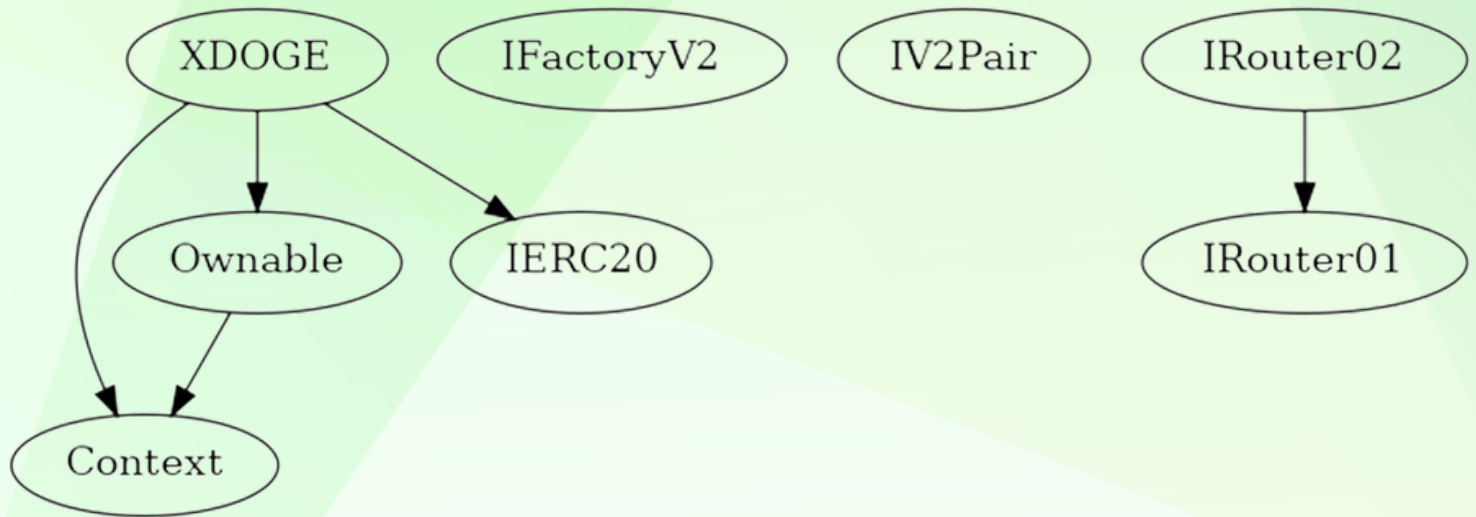
Findings

Severity

Found

◆ Critical	0
◆ High-Risk	1
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0

INHERITANCE TREE





POINTS TO NOTE

- **Owner is able to update buy and sell fees (1% fee)**
 - **Owner is not able to set fee on transfers**
 - Owner is not able to blacklist an arbitrary address.
 - Owner is not able to disable trades
 - Owner is not able to mint new tokens
 - Owner is not able to set maximum wallet and maximum buy/sell limits
-

CONTRACT ASSESMENT














Contract	Type	Bases			
└──────────┘└──────────────────┘└──────────────────┘└──────────────────┘└──────────────────┘					
└┬┘	**Function Name**	**Visibility**	**Mutability**	**Modifiers**	
	Context	Implementation			
└┬┘	<Constructor>	Public !	⊗	NO !	
└┬┘	_msgSender	Internal 🔒			
└┬┘	_msgData	Internal 🔒			
	Ownable	Implementation	Context		
└┬┘	<Constructor>	Public !	⊗	NO !	
└┬┘	owner	Public !		NO !	
└┬┘	renounceOwnership	Public !	⊗	onlyOwner	
└┬┘	transferOwnership	Public !	⊗	onlyOwner	
└┬┘	_setOwner	Private 🔒?	⊗		
	IFactoryV2	Interface			
└┬┘	getPair	External !		NO !	
└┬┘	createPair	External !	⊗	NO !	
	IV2Pair	Interface			
└┬┘	factory	External !		NO !	
└┬┘	getReserves	External !		NO !	
└┬┘	sync	External !	⊗	NO !	
	IRouter01	Interface			
└┬┘	factory	External !		NO !	
└┬┘	WETH	External !		NO !	
└┬┘	addLiquidityETH	External !	ⓈⓉ	NO !	
└┬┘	addLiquidity	External !	⊗	NO !	
└┬┘	swapExactETHForTokens	External !	ⓈⓉ	NO !	
└┬┘	getAmountsOut	External !		NO !	
└┬┘	getAmountsIn	External !		NO !	
	IRouter02	Interface	IRouter01		
└┬┘	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	⊗	NO !	





CONTRACT ASSESMENT

```
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ! | SD | NO !  
|  
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! |  |  
| NO ! |  
| L | swapExactTokensForTokens | External ! |  | NO ! |  
| | | | |  
| **IERC20** | Interface | | | |  
| L | totalSupply | External ! | | NO ! |  
| L | decimals | External ! | | NO ! |  
| L | symbol | External ! | | NO ! |  
| L | name | External ! | | NO ! |  
| L | getOwner | External ! | | NO ! |  
| L | balanceOf | External ! | | NO ! |  
| L | transfer | External ! |  | NO ! |  
| L | allowance | External ! | | NO ! |  
| L | approve | External ! |  | NO ! |  
| L | transferFrom | External ! |  | NO ! |  
| | | | |  
| **XDOGE** | Implementation | Context, Ownable, IERC20 | | |  
| L | totalSupply | External ! | | NO ! |  
| L | decimals | External ! | | NO ! |  
| L | symbol | External ! | | NO ! |  
| L | name | External ! | | NO ! |  
| L | getOwner | External ! | | NO ! |  
| L | allowance | External ! | | NO ! |  
| L | balanceOf | Public ! | | NO ! |  
| L | <Constructor> | Public ! |  | NO ! |  
| L | <Receive Ether> | External ! | SD | NO ! |  
| L | transfer | Public ! |  | NO ! |  
| L | approve | External ! |  | NO ! |  
| L | _approve | Internal  |  | |  
| L | transferFrom | External ! |  | NO ! |  
| L | isNoFeeWallet | External ! | | NO ! |  
| L | setNoFeeWallet | Public ! |  | onlyOwner |  
| L | isLimitedAddress | Internal  | | |
```

CONTRACT ASSESMENT

^L	is_buy	Internal 		
^L	is_sell	Internal 		
^L	canSwap	Internal 		
^L	changeLpPair	External !		onlyOwner
^L	toggleCanSwapFees	External !		onlyOwner
^L	_transfer	Internal 		
^L	changeWallets	External !		onlyOwner
^L	takeTaxes	Internal 		
^L	internalSwap	Internal 		inSwapFlag
^L	setPresaleAddress	External !		onlyOwner
^L	enableTrading	External !		onlyOwner

Legend

Symbol	Meaning
	Function can modify state
	Function is payable



STATIC ANALYSIS

```
Low level call in XDOGE.internalSwap(uint256) (contracts/Token.sol#465-493):
- (success,None) = marketingAddress.call{gas: 35000,value: address(this).balance}() (contracts/Token.sol#488-491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter01.WETH() (contracts/Token.sol#94) is not in mixedCase
Event XDOGE_enableTrading() (contracts/Token.sol#274) is not in CapWords
Event XDOGE_setPresaleAddress(address,bool) (contracts/Token.sol#275) is not in CapWords
Event XDOGE_toggleCanSwapFees(bool) (contracts/Token.sol#276) is not in CapWords
Event XDOGE_changePair(address) (contracts/Token.sol#277) is not in CapWords
Event XDOGE_changeThreshold(uint256) (contracts/Token.sol#278) is not in CapWords
Event XDOGE_changeWallets(address) (contracts/Token.sol#279) is not in CapWords
Function XDOGE.is_buy(address,address) (contracts/Token.sol#371-374) is not in mixedCase
Function XDOGE.is_sell(address,address) (contracts/Token.sol#376-379) is not in mixedCase
Parameter XDOGE.takeTaxes(address,bool,bool,uint256,bool)._lauchtax (contracts/Token.sol#449) is not in mixedCase
Constant XDOGE_totalSupply (contracts/Token.sol#246) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.swapThreshold (contracts/Token.sol#247) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.buyfee (contracts/Token.sol#248) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.sellfee (contracts/Token.sol#249) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.transferfee (contracts/Token.sol#250) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.botFee (contracts/Token.sol#251) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE.fee_denominator (contracts/Token.sol#253) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE_name (contracts/Token.sol#259) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE_symbol (contracts/Token.sol#260) is not in UPPER_CASE_WITH_UNDERSCORES
Constant XDOGE_decimals (contracts/Token.sol#261) is not in UPPER_CASE_WITH_UNDERSCORES
Variable XDOGE.AntiMEV (contracts/Token.sol#265) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#18)" inContext (contracts/Token.sol#10-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#108) is too similar to
IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

XDOGE.AntiMEV (contracts/Token.sol#265) is never used in XDOGE (contracts/Token.sol#201-510)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

XDOGE.AntiMEV (contracts/Token.sol#265) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

XDOGE.lpPair (contracts/Token.sol#263) should be immutable
XDOGE.swapRouter (contracts/Token.sol#258) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output



FUNCTIONAL TESTING

1- Adding liquidity (passed):

<https://testnet.bscscan.com/tx/0x850e035cd38dcad117bfcd030e9c8cef08fdb37eb3f8b5471868e09169dd2d44>

2- Buying when excluded (0% tax) (passed):

<https://testnet.bscscan.com/tx/0x2d815b4afea02b621b49bd5bc77eb4a10ab431efbef5a43aaf6d40a903d7a4d8>

3- Selling when excluded (0% tax) (passed):

<https://testnet.bscscan.com/tx/0xaa3f4959fed8871c7866d066d62eb21e2d5bdade8a26ab40f0f1675cb7d025b7>

4- Transferring when excluded from fees (0% tax) (passed):

<https://testnet.bscscan.com/tx/0x510eb71eefc33b34d79dbb6dd7cfa55f2e50fdb507e99facb71114a579912e6d>

5- Buying when not excluded from fees (1% tax) (passed):

<https://testnet.bscscan.com/tx/0x7ddc9529885c74df56394d87cf200cb278a80e03f2f6d16671fa365f003354ed>

6- Selling when not excluded from fees (1% tax) (passed):

<https://testnet.bscscan.com/tx/0x8b2b28f951d647d7550f1fd206b1f42b6b14e5e6547f714482e9a6845cc35e1c>



FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):

<https://testnet.bscscan.com/tx/0x3e8235c1c061e208686b9c1051edc040e45055ede7c1216e6ea75383e57ba448>

7- Internal swap (passed):

<https://testnet.bscscan.com/address/0xdC447192E935202FBF4A8Aff3FF95649b4Cd2dA4#internaltx>

High Risk

Centralization – swaps are disabled by default

Severity: **High**

function: enableTrading

Status: **Resolved (Trades are enabled)**

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function enableTrading(uint256 deadline) external onlyOwner {
    require(deadline < 5, "Deadline too high");
    require(!isTradingEnabled, "Trading already enabled");
    isTradingEnabled = true;
    _deadline = block.number + deadline;
    emit _enableTrading();
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades



DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.



ABOUT AUDITACE

We specialize in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.



<https://auditace.tech/>



https://t.me/Audit_Ace



https://twitter.com/auditace_



<https://github.com/Audit-Ace>
