AUDIT REPORT
SecureWise

DOGECORE

https://securewise.info/ https://t.me/securewisehub https://github.com/securewise

Table of Contents

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

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

Overview

Token Name: DogeCore **(DCORE)**

Methodology: Automated Analysis, Manual Code Review

Language: Solidity

Contract Address: 0x8063F3ff48B24cAe82DbA04D24D11b8A3B9A087c

ContractLink: https://scan.coredao.org/address/0x8063F3ff48B24cAe82DbA04D24D11b8A3B9A087c

Network: Core

Supply: 100.000.000

Website: https://www.dogecore.xyz/

Twitter: https://twitter.com/DogeCoreTW

Telegram: https://t.me/dogecoreportal

Report Date: March 1, 2023

Quick Result

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

⚠	The owner can set fees within reasonable limits
⚠	The owner can exclude accounts from fees
⚠	The owner can set max transaction amount within reasonable limits
⚠	The owner can change swap settings

DogeCore (DCORE) has succesfully **PASSED** the smart contract audit with **LOW** severity issue

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

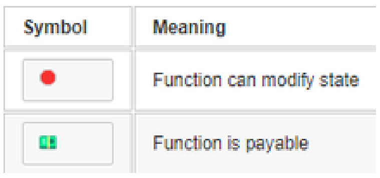
Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

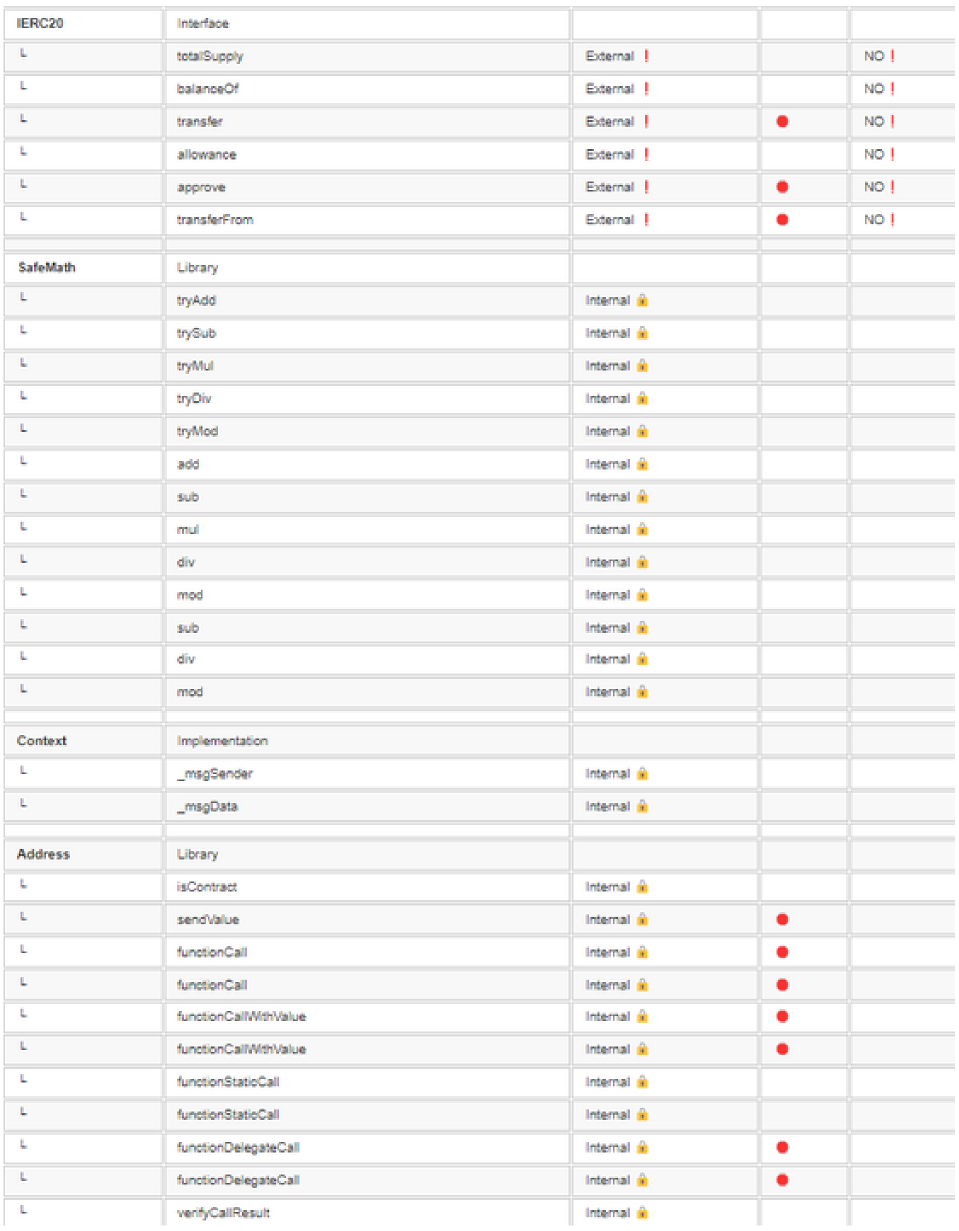
Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.

Low: Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.

Automated Analysis

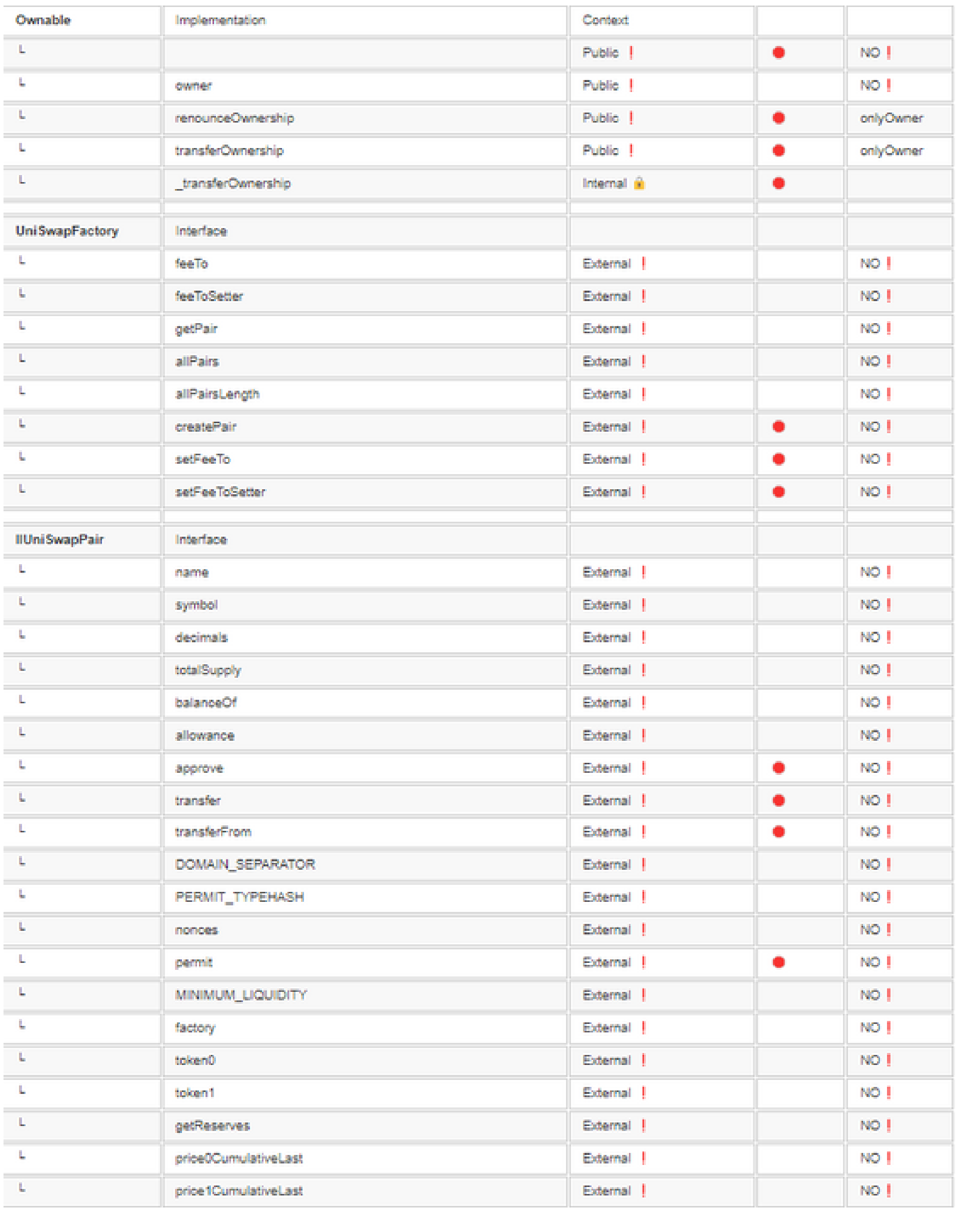


Symbol	Meaning
●	Function can modify state
💵	Function is payable



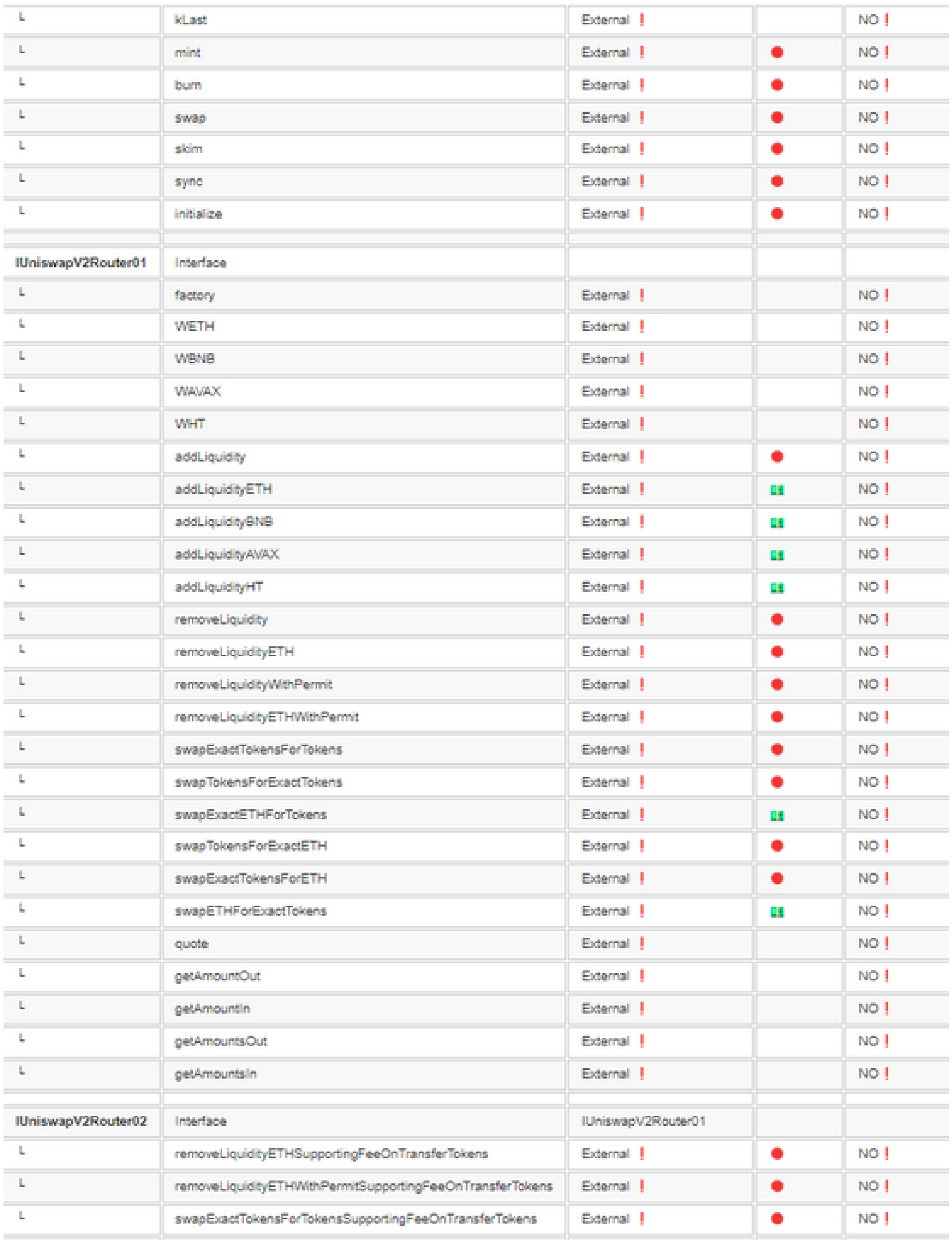
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	●	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	●	NO ❗
└	transferFrom	External ❗	●	NO ❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	●	
└	functionCall	Internal 🔒	●	
└	functionCall	Internal 🔒	●	
└	functionCallWithValue	Internal 🔒	●	
└	functionCallWithValue	Internal 🔒	●	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	●	
└	functionDelegateCall	Internal 🔒	●	
└	verifyCallResult	Internal 🔒		

Automated Analysis



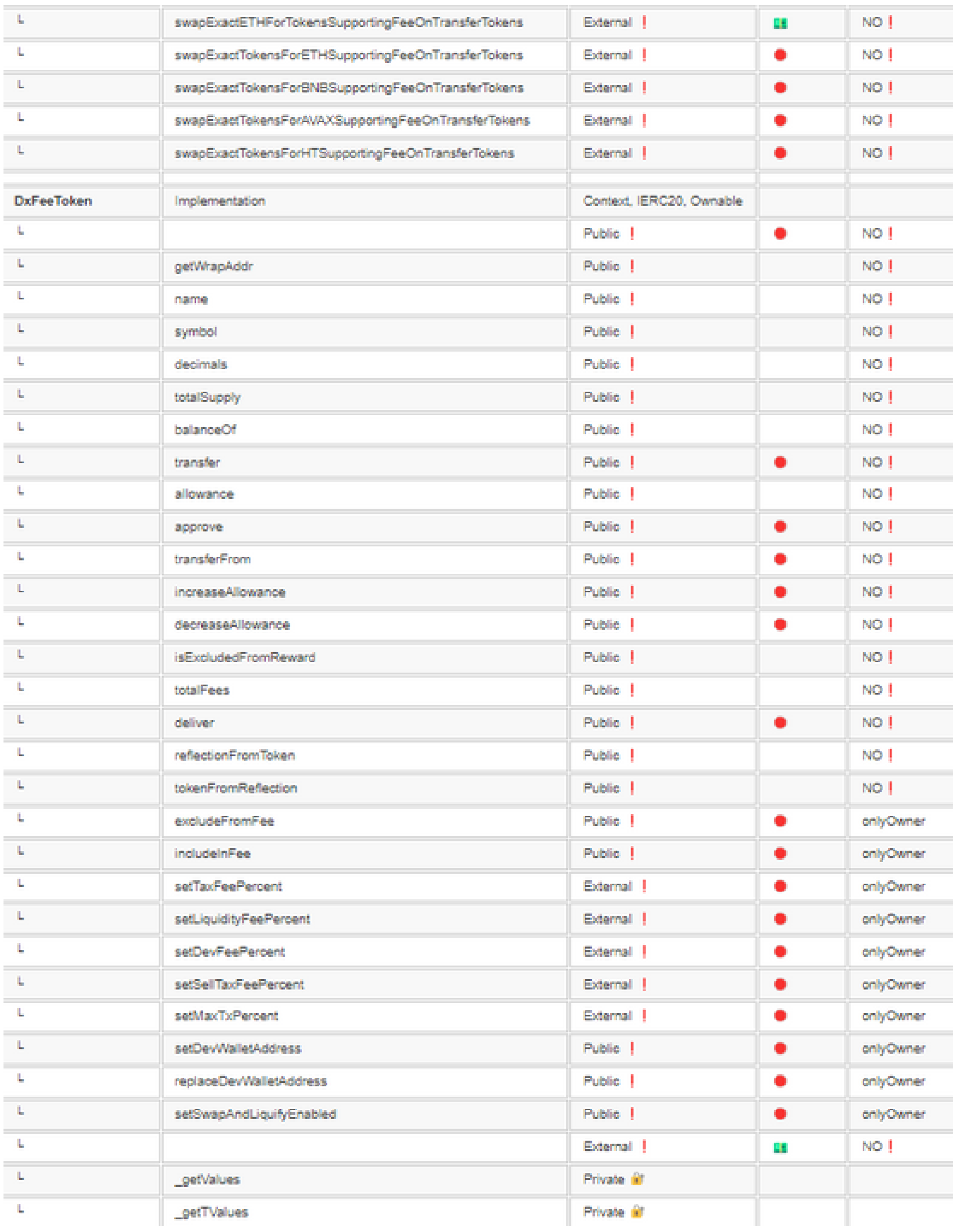
Ownable	Implementation	Context		
└		Public ❗	●	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	●	onlyOwner
└	transferOwnership	Public ❗	●	onlyOwner
└	_transferOwnership	Internal 🔒	●	
UniSwapFactory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗
└	allPairsLength	External ❗		NO ❗
└	createPair	External ❗	●	NO ❗
└	setFeeTo	External ❗	●	NO ❗
└	setFeeToSetter	External ❗	●	NO ❗
IUniSwapPair	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	●	NO ❗
└	transfer	External ❗	●	NO ❗
└	transferFrom	External ❗	●	NO ❗
└	DOMAIN_SEPARATOR	External ❗		NO ❗
└	PERMIT_TYPEHASH	External ❗		NO ❗
└	nonces	External ❗		NO ❗
└	permit	External ❗	●	NO ❗
└	MINIMUM_LIQUIDITY	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	token0	External ❗		NO ❗
└	token1	External ❗		NO ❗
└	getReserves	External ❗		NO ❗
└	price0CumulativeLast	External ❗		NO ❗
└	price1CumulativeLast	External ❗		NO ❗

Automated Analysis



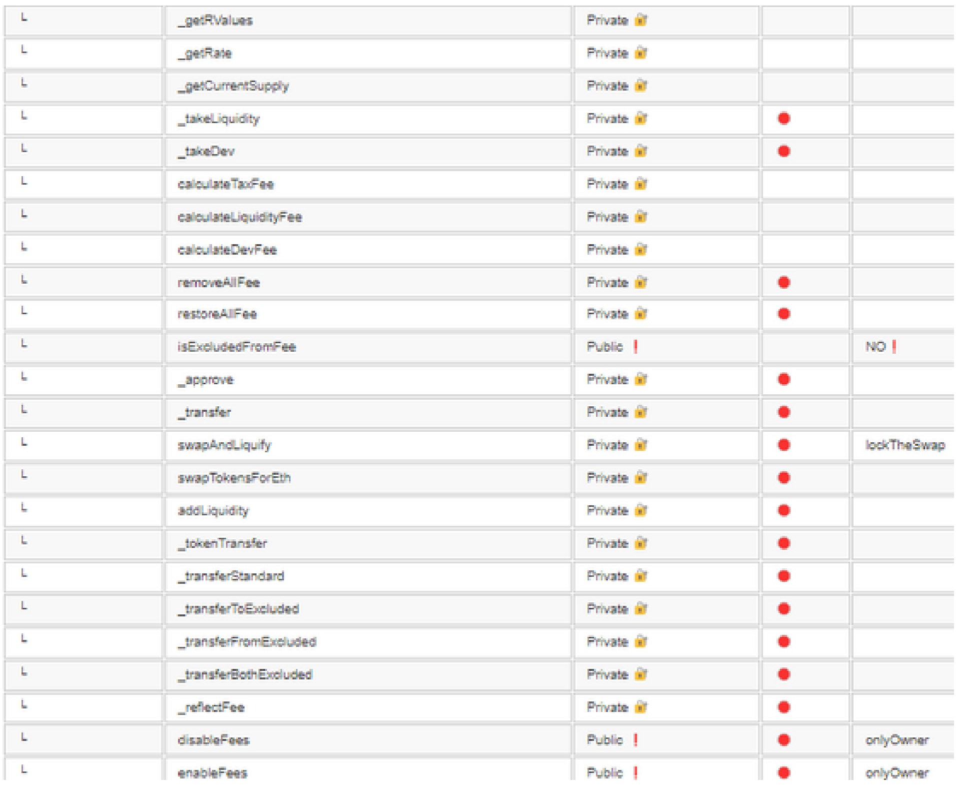
└	kLast	External ❗		NO ❗
└	mint	External ❗	●	NO ❗
└	burn	External ❗	●	NO ❗
└	swap	External ❗	●	NO ❗
└	skim	External ❗	●	NO ❗
└	sync	External ❗	●	NO ❗
└	initialize	External ❗	●	NO ❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	WBNB	External ❗		NO ❗
└	WAVAX	External ❗		NO ❗
└	WHT	External ❗		NO ❗
└	addLiquidity	External ❗	●	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	addLiquidityBNB	External ❗	💵	NO ❗
└	addLiquidityAVAX	External ❗	💵	NO ❗
└	addLiquidityHT	External ❗	💵	NO ❗
└	removeLiquidity	External ❗	●	NO ❗
└	removeLiquidityETH	External ❗	●	NO ❗
└	removeLiquidityWithPermit	External ❗	●	NO ❗
└	removeLiquidityETHWithPermit	External ❗	●	NO ❗
└	swapExactTokensForTokens	External ❗	●	NO ❗
└	swapTokensForExactTokens	External ❗	●	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapTokensForExactETH	External ❗	●	NO ❗
└	swapExactTokensForETH	External ❗	●	NO ❗
└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	●	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	●	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	●	NO ❗

Automated Analysis



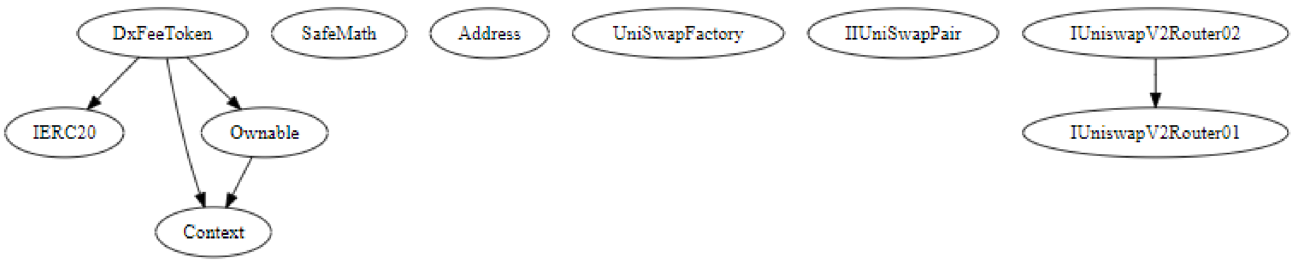
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForBNBSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForAVAXSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForHTSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
DxFeeToken	Implementation	Context, IERC20, Ownable		
└		Public ❗	🛑	NO ❗
└	getWrapAddr	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 ❗
└	isExcludedFromReward	Public ❗		NO ❗
└	totalFees	Public ❗		NO ❗
└	deliver	Public ❗	🛑	NO ❗
└	reflectionFromToken	Public ❗		NO ❗
└	tokenFromReflection	Public ❗		NO ❗
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	setTaxFeePercent	External ❗	🛑	onlyOwner
└	setLiquidityFeePercent	External ❗	🛑	onlyOwner
└	setDevFeePercent	External ❗	🛑	onlyOwner
└	setSellTaxFeePercent	External ❗	🛑	onlyOwner
└	setMaxTxPercent	External ❗	🛑	onlyOwner
└	setDevWalletAddress	Public ❗	🛑	onlyOwner
└	replaceDevWalletAddress	Public ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└		External ❗	💵	NO ❗
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		

Automated Analysis

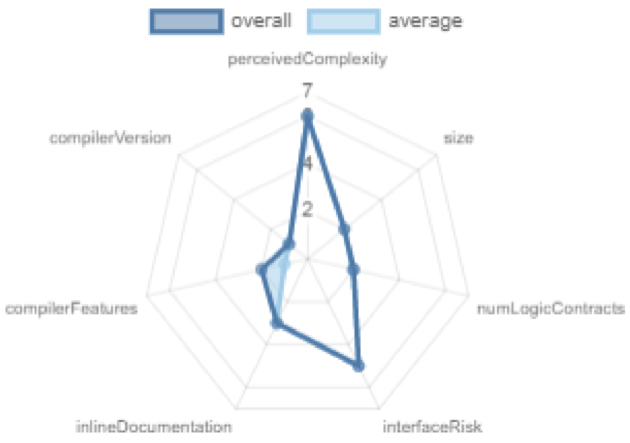


└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	●	
└	_takeDev	Private 🔐	●	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	calculateDevFee	Private 🔐		
└	removeAllFee	Private 🔐	●	
└	restoreAllFee	Private 🔐	●	
└	isExcludedFromFee	Public ❗		NO ❗
└	_approve	Private 🔐	●	
└	_transfer	Private 🔐	●	
└	swapAndLiquify	Private 🔐	●	lockTheSwap
└	swapTokensForEth	Private 🔐	●	
└	addLiquidity	Private 🔐	●	
└	_tokenTransfer	Private 🔐	●	
└	_transferStandard	Private 🔐	●	
└	_transferToExcluded	Private 🔐	●	
└	_transferFromExcluded	Private 🔐	●	
└	_transferBothExcluded	Private 🔐	●	
└	_reflectFee	Private 🔐	●	
└	disableFees	Public ❗	●	onlyOwner
└	enableFees	Public ❗	●	onlyOwner

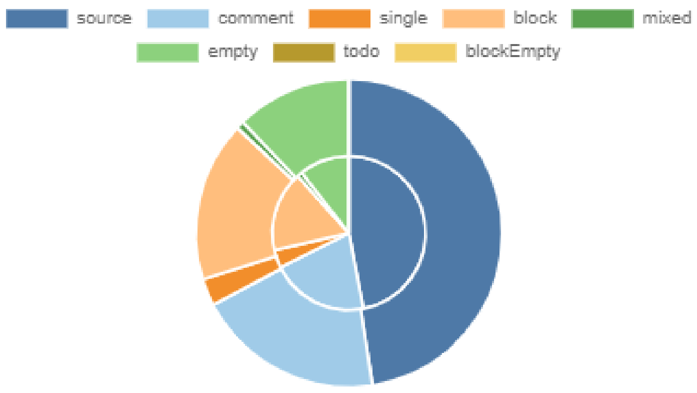
Inheritance Graph



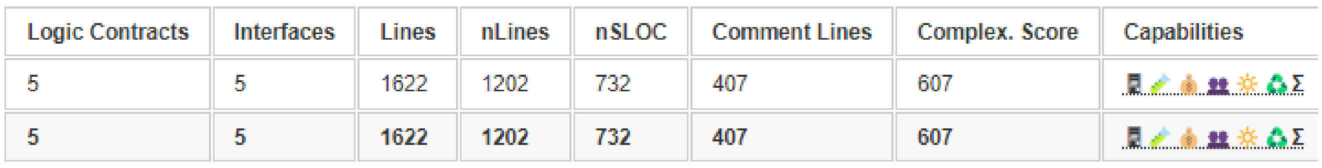
Risk



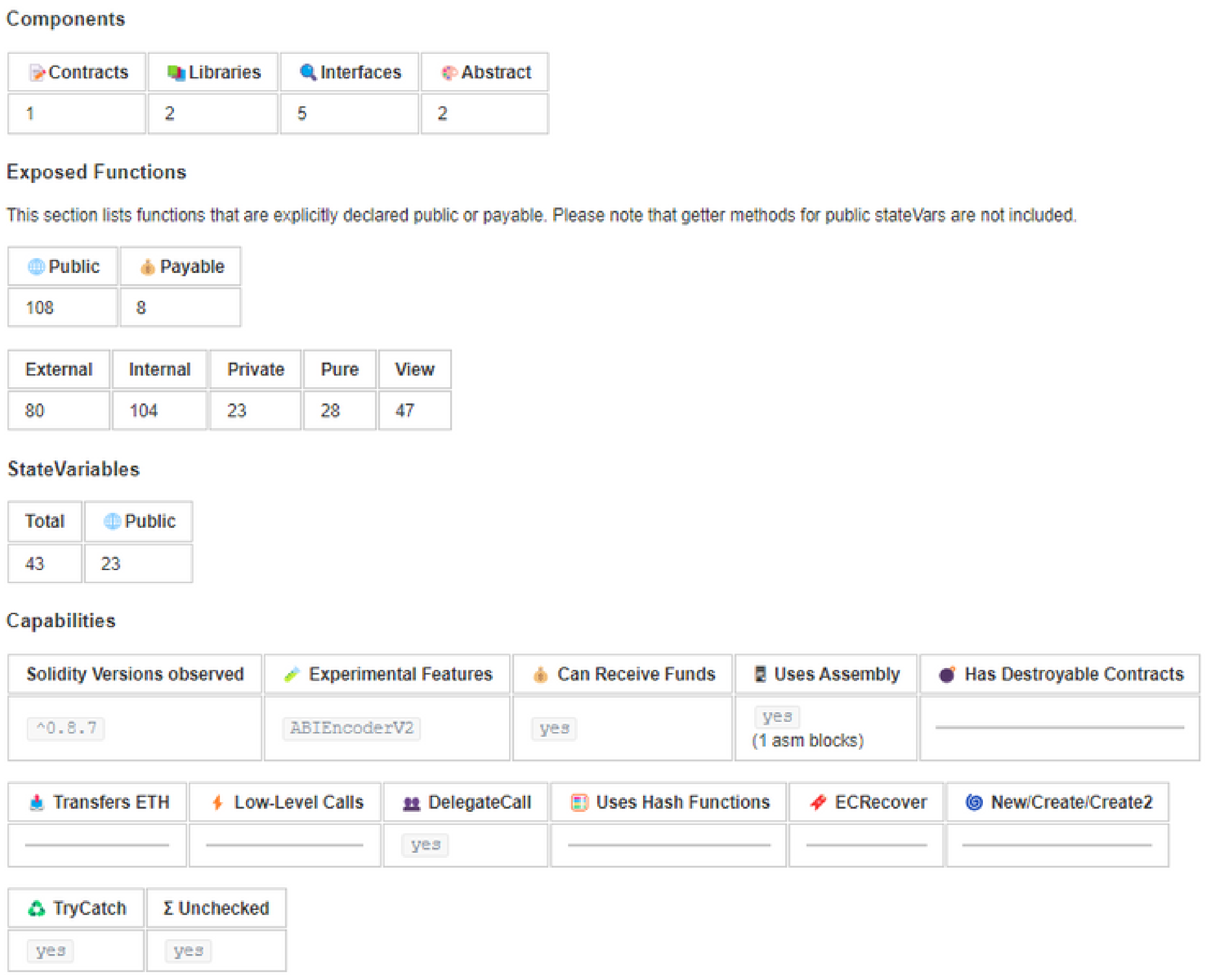
Source Lines



Contract Summary



Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
5	5	1622	1202	732	407	607	🖥🧪💰👥☀️♻️Σ
5	**5**	**1622**	**1202**	**732**	**407**	**607**	🖥🧪💰👥☀️♻️Σ



Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
1	2	5	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
108	8

External	Internal	Private	Pure	View
80	104	23	28	47

StateVariables

Total	🌐Public
43	23

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.7`	`ABIEncoderV2`	`yes`	`yes` (1 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
		`yes`			

♻️ TryCatch	Σ Unchecked
`yes`	`yes`

Manual Review

Unchecked return value

```
   uniswapV2Router.addLiquidityETH{value   :
ETHAmount}(
              address(this),
              tokenAmount,
              0, // slippage is unavoidable
              0, // slippage is unavoidable
              dead,
              block.timestamp
            );
```

If the return value of a low-level call is not checked, the execution may resume even if the function call throws an error. This can lead to unexpected behaviour and break the program logic. A failed call can even be caused by an attacker, who may be able to further exploit the contract.

Recommendation

In the case that you use low-level calls, be sure to check the return value to handle possible failed calls.

Manual Review

Lacks a zero-check on set wallets function

```
DxFeeToken.constructor(....)
router = _router;
_devWalletAddress = devWalletAddress_;
basePair = _basePair;
```

Zero-address checks as input validation on address parameters is always a best practice. This is especially true for critical addresses that are immutable and set in the constructor because they cannot be changed later. Accidentally using zero addresses here will lead to failing logic or force contract redeployment and increased gas costs.

Recommendation

Add zero-address input validation for these addresses.

Manual Review

Access Modifiers Vulnerabilities

```
mintedByDxsale
transferFrom()
totalFees()
renounceOwnership()
setDevWalletAddress()
replaceDevWalletAddress()
enableFees()
disableFees()
isExcludedFromFee()
transferOwnership()
setSwapAndLiquifyEnabled()
```

These functions are used as public instead of external.

Recommendation

Access control identifiers must be authenticated and set adequately to avoid possible vulnerabilities

Out date compiler version

```
pragma solidity ^0.8.7;
```

Compiler is set an outdated version.

Recommendation

Set and use new versions

Floating Pragma

```
pragma solidity ^0.8.7;
```

Recommendation

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

AUDIT REPORT

SecureWise