AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

CocksGG

DATED : 6 august 23'

AUDIT SUMMARY

Project name –CocksGG

Date: 6 august, 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	0	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/token/0x760AC5CBAAC7a539498333faC18D098e7e92aaC0

Token Information

Token Name : CocksGG

Token Symbol: COGG

Decimals: 18

Token Supply: 100,000,000

Token Address:
0xeF76b289F5c925395379Ee950Ec1daF44f812af5

Checksum:
7eba8e1c5a96b475873b54d016531c1948c2cadc

Owner:
0xC58FFC041806c4FFE34943C27703A3a7FdA5baEf
(at time of writing the audit)

Deployer:
0x4942dFd6d7c9466B4a0Df1B8aB8B2728b9d26301

TOKEN OVERVIEW

Fees:

Buy Fees:0-3%

Sell Fees: 0-3%

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

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

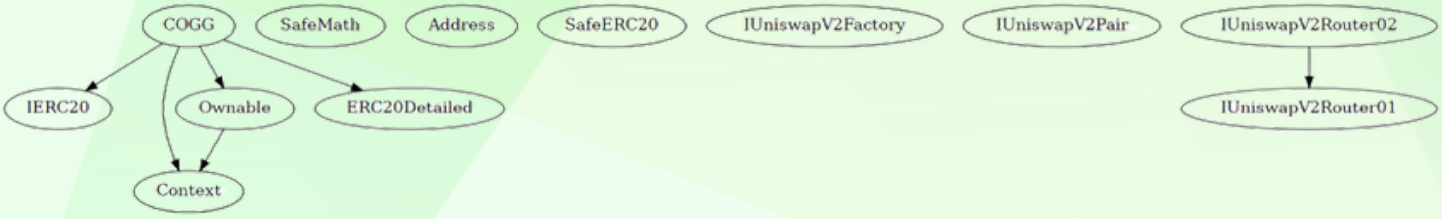
CLASSIFICATION OF RISK

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	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to update buy/sell fees within 0-3%
- Owner is not able to set fees on transfers
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to mint new tokens
- Owner is not able to set max buy/sell/transfer

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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			
└	<Constructor>	Public ❗	🛑	NO ❗
└	_msgSender	Internal 🔒		

CONTRACT ASSESMENT

Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
ERC20Detailed	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗	NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
Address	Library			
└	isContract	Internal 🔒		
SafeERC20	Library			
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeApprove	Internal 🔒	🛑	
└	callOptionalReturn	Private 🔐	🛑	
IUniswapV2Factory	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 ❗
IUniswapV2Pair	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 ❗

CONTRACT ASSESMENT

└	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 ❗
└	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 ❗
└	addLiquidity	External ❗	🛑 |NO ❗
└	addLiquidityETH	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 ❗

||||||

CONTRACT ASSESMENT

IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
COGG	Implementation	Context, Ownable, IERC20, ERC20Detailed		
└	<Constructor>	Public ❗	🛑	ERC20Detailed
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_approve	Internal 🔒	🛑	
└	isContract	Internal 🔒		
└	setBuyMarketingFeePercent	External ❗	🛑	onlyOwner
└	setSellMarketingFeePercent	External ❗	🛑	onlyOwner
└	setMarketingAddress	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	changeNumTokensSellToFee	External ❗	🛑	onlyOwner
└	clearETH	External ❗	🛑	onlyOwner
└	clearERC20	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO ❗
└	<Receive Ether>	External ❗	💵	NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in COGG._transfer(address,address,uint256) (contracts/Token.sol#833-894):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/Token.sol#856)
                - address(marketingAddress).transfer(address(this).balance) (contracts/Token.sol#899)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (contracts/Token.sol#878-881)
        - _balances[recipient] = _balances[recipient].add(TotalSent) (contracts/Token.sol#882)
        - _balances[address(this)] = _balances[address(this)].add(taxAmount) (contracts/Token.sol#883)
        - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (contracts/Token.sol#887-890)
        - _balances[recipient] = _balances[recipient].add(amount) (contracts/Token.sol#891)
        - marketingFee = buyMarketingFee (contracts/Token.sol#872)
        - marketingFee = sellMarketingFee (contracts/Token.sol#874)
        Event emitted after the call(s):
        - Transfer(sender,recipient,TotalSent) (contracts/Token.sol#884)
        - Transfer(sender,address(this),taxAmount) (contracts/Token.sol#885)
        - Transfer(sender,recipient,amount) (contracts/Token.sol#892)
Reentrancy in COGG.swapAndLiquify(uint256) (contracts/Token.sol#896-901):
        External calls:
        - address(marketingAddress).transfer(address(this).balance) (contracts/Token.sol#899)
        Event emitted after the call(s):
        - SwapAndLiquify(contractTokenBalance,address(this).balance) (contracts/Token.sol#900)
Reentrancy in COGG.transferFrom(address,address,uint256) (contracts/Token.sol#702-717):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#707)
                - address(marketingAddress).transfer(address(this).balance) (contracts/Token.sol#899)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#708-715)
                - _allowances[towner][spender] = amount (contracts/Token.sol#749)
        Event emitted after the call(s):
        - Approval(towner,spender,amount) (contracts/Token.sol#750)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#708-715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#420) is too similar to IUniswapV2Router01.addLiquidity(addr
ess,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
COGG.constructor() (contracts/Token.sol#645-669) uses literals with too many digits:
        - _totalSupply = 100000000 * (10 ** 18) (contracts/Token.sol#647)
COGG.changeNumTokensSellToFee(uint256) (contracts/Token.sol#792-802) uses literals with too many digits:
        - require(bool,string)(_numTokensSellToFee >= 10000 * 10 ** 18 && _numTokensSellToFee <= 1000000 * 10 ** 18,Swap to fee threshold must be set within 10,000 to 1,000,000 tokens) (contracts/Token
.sol#795-799)
COGG.slitherConstructorVariables() (contracts/Token.sol#608-920) uses literals with too many digits:
        - numTokensSellToFee = 100000 * 10 ** 18 (contracts/Token.sol#631)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
COGG._owner (contracts/Token.sol#643) should be immutable
COGG._totalSupply (contracts/Token.sol#620) 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/0xf5d00a9ec174768ff1dc54f6d988e215b39dfe1bfbaad93b2e120f5d689d866c

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

https://testnet.bscscan.com/tx/0xc24f92460100142965400fe476d338cbda174c09f331f811fbc4571493a825ab

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

https://testnet.bscscan.com/tx/0x46c61402f7cbcea140d1e1858db85bb9f9f89fdee2dd27aa077949953e22d163

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

https://testnet.bscscan.com/tx/0x792b4646fea598d9bb807b1a0da23c87a7af565cf55b3484e59c1a49e44fcf3a

5- Buying when not excluded from fees (0-3% tax) (passed):

https://testnet.bscscan.com/tx/0xf49a59494a776f10e38adbb2a0114a2aeefdd6ed9c9697b2c01b08d82321a722

6- Selling when not excluded from fees (0-3% tax) (passed):

https://testnet.bscscan.com/tx/0xd64253c98b1b070f292caa01e86b519281edc3d5bd8135b9eb3eab2b9d80a313

FUNCTIONAL TESTING

7- Transferring (0% tax) (passed):

https://testnet.bscscan.com/tx/0x2fe848ee4b1e6c545049f27a43c361c0a301ff1fec9b51c8f7156c3718a557df

8- Internal swap (BNB sent to marketing wallet) (passed):

https://testnet.bscscan.com/tx/0xd64253c98b1b070f292caa01e86b519281edc3d5bd8135b9eb3eab2b9d80a313

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 specializes 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