auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Floki CFO

DATED : 28 April 23'

AUDIT SUMMARY

Project name – Floki CFO

Date: 28 April, 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	1	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/token/0x6a86431caa88145cc62d17d77b380e14c66fefaa

Token Information

Token Name : Floki CFO

Token Symbol: Floki CFO

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x8FB834bDb8B940493771274D0581Cee0bc1E7387
Checksum:
dc79657f9a05c62539ce3f14295ca67fdceb31db

Owner:
0xa6a8a4c9AAf02bF15A4386821FDA61A99A58437a
(at the time of writing the audit)

Deployer:

0xa6a8a4c9AAf02bF15A4386821FDA61A99A58437a

TOKEN OVERVIEW

Fees:

Buy Fees: up to 10%

Sell Fees: up to 10%

Transfer Fees: up to 10%

Fees Privilege: Owner

Ownership: Owned by safu dev for 14 days

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: modifying swap threshold - toggling internal swap - excluding wallets from fee - including wallets in fee - 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

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

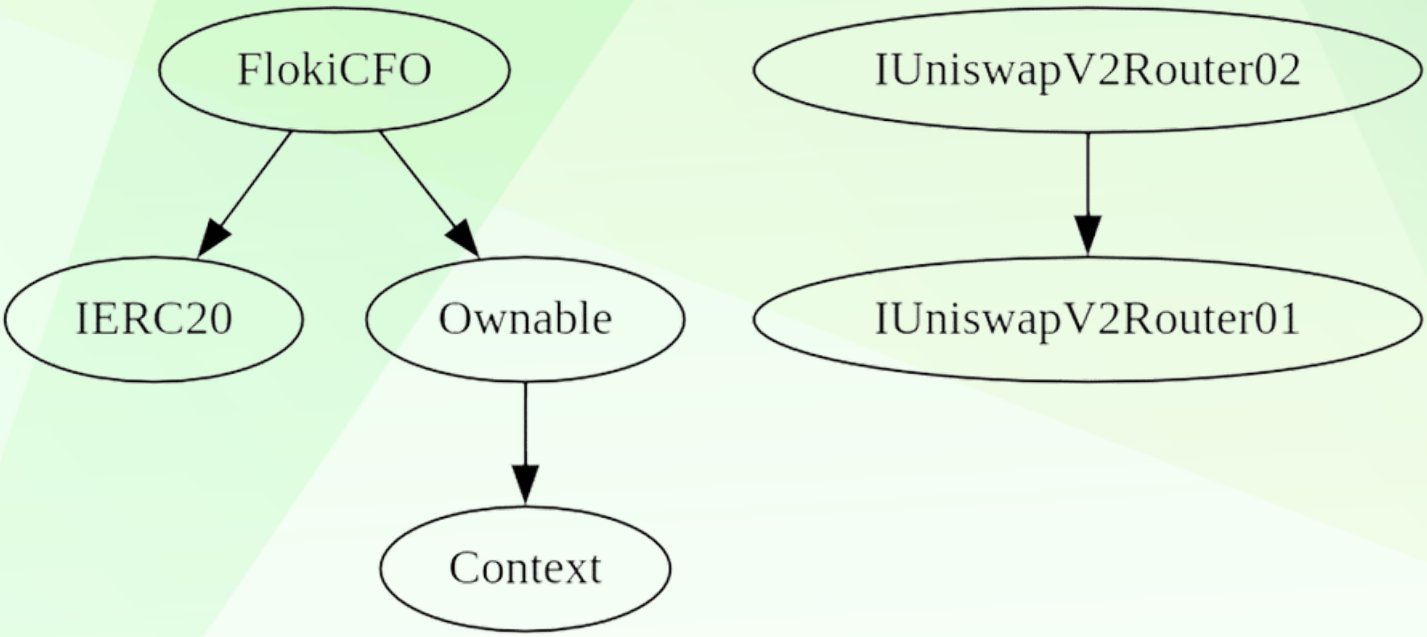
CLASSIFICATION OF RISK

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	1
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to change buy/sell/transfer each one up to 10%
- Owner is not able to change fees until 7 days after launch
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner has to enable trades manually for holders
- Owner is not able to mint new tokens
- Token has FlokiCEO and RedFloki rewards with auto-distribution
- Fees are used for rewards and marketing

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
FlokiCFO	Implementation	IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	<Receive Ether>	External ❗	💵	NO ❗
└	totalSupply	External ❗		NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	_approve	Internal 🔒	🛑	
└	approveMax	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	_transferFrom	Internal 🔒	🛑	
└	takeFee	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		
└	shouldDoContractSwap	Internal 🔒		
└	isRewardExcluded	Public ❗		NO ❗
└	isFeeExcluded	Public ❗		NO ❗
└	doContractSwap	Internal 🔒	🛑	swapping
└	swapTokensForTokens	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	setIsDividendExempt	External ❗	🛑	onlyOwner
└	setIsFeeExempt	External ❗	🛑	onlyOwner
└	setDoContractSwap	External ❗	🛑	onlyOwner
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	setDistributorSettings	External ❗	🛑	onlyOwner
└	changeMarketingWallet	External ❗	🛑	onlyOwner
└	changeBuyFees	External ❗	🛑	onlyOwner
└	changeSellFees	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	setAuthorizedWallets	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	changeGetFeesOnTransfer	External ❗	🛑	onlyOwner
└	changeRouter	External ❗	🛑	onlyOwner
└	changePair	External ❗	🛑	onlyOwner

CONTRACT ASSESMENT

Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	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 ❗

CONTRACT ASSESMENT

└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
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 ❗
IDividendDistributor	Interface			
└	setDistributionCriteria	External ❗	🛑	NO ❗
└	setShare	External ❗	🛑	NO ❗
└	deposit	External ❗	🛑	NO ❗
└	process	External ❗	🛑	NO ❗
└	purge	External ❗	🛑	NO ❗
DividendDistributor	Implementation	IDividendDistributor		
└	<Constructor>	Public ❗	🛑	NO ❗
└	<Receive Ether>	External ❗	💵	NO ❗
└	setDistributionCriteria	External ❗	🛑	onlyToken
└	purge	External ❗	🛑	onlyToken
└	setShare	External ❗	🛑	onlyToken
└	deposit	External ❗	🛑	onlyToken
└	process	External ❗	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	
└	claimDividend	External ❗	🛑	NO ❗
└	getUnpaidEarnings	Public ❗		NO ❗
└	getHolderDetails	Public ❗		NO ❗
└	getCumulativeDividends	Internal 🔒		
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfTokenHolders	External ❗		NO ❗
└	getShareHoldersList	External ❗		NO ❗

CONTRACT ASSESMENT

└	totalDistributedRewards	External ❗			NO ❗
└	addShareholder	Internal 🔒	🛑		
└	removeShareholder	Internal 🔒	🛑		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in FlokiCFO._transferFrom(address,address,uint256) (contracts/fceo/Untitled-1.sol#886-946):
        External calls:
        - doContractSwap() (contracts/fceo/Untitled-1.sol#898)
                - address(marketingWallet).transfer(swappedTokens) (contracts/fceo/Untitled-1.sol#1050)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender] - amount (contracts/fceo/Untitled-1.sol#902)
        - _balances[recipient] = _balances[recipient] + amountReceived (contracts/fceo/Untitled-1.sol#907)
        - amountReceived = takeFee(sender,recipient,amount) (contracts/fceo/Untitled-1.sol#904-906)
                - _balances[address(this)] = _balances[address(this)] + feeToken (contracts/fceo/Untitled-1.sol#958)
        Event emitted after the call(s):
        - Transfer(sender,address(this),feeToken) (contracts/fceo/Untitled-1.sol#959)
                - amountReceived = takeFee(sender,recipient,amount) (contracts/fceo/Untitled-1.sol#904-906)
        - Transfer(sender,recipient,amountReceived) (contracts/fceo/Untitled-1.sol#944)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/fceo/Untitled-1.sol#100) is too similar to IUniswapV2Router01.a
ddLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/fceo/Untitled-1.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

FlokiCFO.slitherConstructorVariables() (contracts/fceo/Untitled-1.sol#700-1212) uses literals with too many digits:
        - _totalSupply = 1000000000 * (10 ** _decimals) (contracts/fceo/Untitled-1.sol#711)
FlokiCFO.slitherConstructorVariables() (contracts/fceo/Untitled-1.sol#700-1212) uses literals with too many digits:
        - distributorGas = 500000 (contracts/fceo/Untitled-1.sol#742)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

DividendDistributor.initialized (contracts/fceo/Untitled-1.sol#319) is never used in DividendDistributor (contracts/fceo/Untitled-1.sol#291-507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DividendDistributor.dividendsPerShareAccuracyFactor (contracts/fceo/Untitled-1.sol#312) should be constant
DividendDistributor.initialized (contracts/fceo/Untitled-1.sol#319) should be constant
FlokiCFO.DEAD (contracts/fceo/Untitled-1.sol#701) should be constant
FlokiCFO.FLOKICEO (contracts/fceo/Untitled-1.sol#705) should be constant
FlokiCFO.REDFLOKI (contracts/fceo/Untitled-1.sol#704) should be constant
FlokiCFO.ZERO (contracts/fceo/Untitled-1.sol#702) should be constant
FlokiCFO._totalSupply (contracts/fceo/Untitled-1.sol#711) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DividendDistributor.REWARD (contracts/fceo/Untitled-1.sol#300) should be immutable
DividendDistributor._token (contracts/fceo/Untitled-1.sol#292) should be immutable
FlokiCFO.flokiCEODividendTracker (contracts/fceo/Untitled-1.sol#740) should be immutable
FlokiCFO.redFlockiDividendTracker (contracts/fceo/Untitled-1.sol#739) should be immutable
FlokiCFO.swapThreshold (contracts/fceo/Untitled-1.sol#746) 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/0xd8e7c9d87d4b1d729f174fba8564ac1fa83de1d0a4242676e5167d6ba606cba2

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

https://testnet.bscscan.com/tx/0xea69583efdb3c5f792584eea0c6aff00f91a7c84ad2f8f58c0d348073e34e69a

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

https://testnet.bscscan.com/tx/0x2cce2adb5505938df5017f0b6b4eac40f05614d693a298caffffa21bba6b5010

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

https://testnet.bscscan.com/tx/0xbf69018352809fac05d2c5ab2de090e0fcec5f3d0536d7a18322def9d046f7f2

5- Buying when not excluded from fees(up to 10% tax (7 days after launch)) (passed):

https://testnet.bscscan.com/tx/0x094a778775617b9ce9580618c739ec2ed138bda9adabd295b266614df20fd909

6- Selling when not excluded from fees (up to 10% tax) (passed):

https://testnet.bscscan.com/tx/0x7c2850dafdceded78af5304b3f4b836431fa5efbfcb263e11d42dd30d3c61e1c

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (10% tax) (passed):

https://testnet.bscscan.com/tx/0xfd009357a9973f1a64ee15a598928cd61eb06030e0d51a44a25ac3e499bd88d5

8- Internal swap (passed):

Marketing wallet received BNB

https://testnet.bscscan.com/address/0x2433e36dc7d27606d9e863b5194380e2be42a720#internaltx

9- Rewards Distribution (passed):

Both dividend trackers distributed rewards between holders

https://testnet.bscscan.com/tx/0x7c2850dafdceded78af5304b3f4b836431fa5efbfcb263e11d42dd30d3c61e1c

MANUAL TESTING

Centralization - Owner Must Enable Trades

Severity: High / Informational

Function: enableTrading

Lines: 492

Status: Resolved (Contract is owned by safu developer)

Overview:

The owner is required to enable trading for investors. If trading remains disabled, token holders will not have the ability to buy, sell, or transfer their tokens.

```
function enableTrading() external onlyOwner {
    isTradeEnabled = true;
    listingTime = block.timestamp;
}
```

Recommendation:

While the presence of this function is considered a feature rather than a flaw, it is crucial to highlight the centralization risk it inherently poses. To address this issue and ensure the enablement of trades, one possible solution would be to transfer the contract's ownership to a trusted third party, such as a Pinksale Safu developer. This would help mitigate the centralization risk associated with this function.

MANUAL TESTING

Logical – Lack of withdraw function for ERC20 tokens

Severity: Low

Status: Not Resolved

Overview:

The present contract implementation does not include a function that allows the owner to retrieve ERC20 tokens that have become stranded within the contract.

Recommendation:

Create a function which enables owner to withdraw stuck ERC20 tokens from the contract

```
function rescueTokens(address _token) public onlyOwner {
        IERC20(_token).transfer(msg.sender,
        IERC20(_token).balanceOf(address(this));
}
```

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