AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

SuperFloki

DATED : 17 July 23'

MANUAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTradingEnabled

Status: Not Resolved

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 enableTradingEnabled() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
}
```

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

AUDIT SUMMARY

Project name – SuperFloki

Date: 17 July, 2023

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

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	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/0x5d75de6551adB90f4729eFACa41d00EF971BfAA3

Token Information

Token Name : SuperFloki

Token Symbol: SuperFloki

Decimals: 9

Token Supply: 100,000,000

Token Address:
0xc630Cef57cDC15EC0d9db92C6FB7Aa62fEc94D97

Checksum:
302f47219d8be09b96fd193c6ea654790e191286

Owner:
0x35b4a89c145Ebb971B045187E311DD29da05D9f6
(at time of writing the audit)

Deployer:
0x9F66133b865fF44c2105d5063bB69658E84dc9F3

TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0%

Fees Privilege: no fees

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: yes

Blacklist: No

Other Privileges: Initial distribution of the tokens

- modifying fees
- enabling trades

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	0
Gas Optimization / Suggestions	0

INHERITANCE TREE

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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 ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	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 ❗
└	_transfer	Internal 🔒	🛑	
└	_tokengeneration	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		

CONTRACT ASSESMENT

└	mod	Internal 🔒			
└	mod	Internal 🔒			
SafeMathInt	Library			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	sub	Internal 🔒			
└	add	Internal 🔒			
└	abs	Internal 🔒			
└	toUint256Safe	Internal 🔒			
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒			
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
IPair	Interface			
└	sync	External ❗	🛑	NO ❗
IFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
└	getPair	External ❗		NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO ❗
└	distributeDividends	External ❗	💵	NO ❗
└	withdrawableDividendOf	External ❗		NO ❗
└	withdrawnDividendOf	External ❗		NO ❗
└	accumulativeDividendOf	External ❗		NO ❗
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	distributeDividends	Public ❗	💵	NO ❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	

CONTRACT ASSESMENT

└	setRewardToken	External ❗	🛑	onlyOwner
└	swapBnbForCustomToken	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	withdrawnDividendOf	Public ❗		NO ❗
└	accumulativeDividendOf	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	_tokengeneration	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
IterableMapping	Library			
└	get	Internal 🔒		
└	getIndexOfKey	Internal 🔒		
└	getKeyAtIndex	Internal 🔒		
└	size	Internal 🔒		
└	set	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
Address	Library			
└	sendValue	Internal 🔒	🛑	
SuperFloki	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	processDividendTracker	External ❗	🛑	NO ❗
└	claim	External ❗	🛑	NO ❗
└	rescueBEP20Tokens	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	NO ❗
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	setBuyTaxes	External ❗	🛑	onlyOwner
└	setSellTaxes	External ❗	🛑	onlyOwner
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	enableTradingEnabled	External ❗	🛑	onlyOwner
└	setBotBlocks	External ❗	🛑	onlyOwner
└	setMinBalanceForDividends	External ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	setGasForProcessing	External ❗	🛑	onlyOwner
└	setClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO ❗
└	getTotalDividendsDistributed	External ❗		NO ❗
└	isExcludedFromFees	Public ❗		NO ❗

CONTRACT ASSESMENT

└	withdrawableDividendOf	Public ❗		NO ❗
└	getCurrentRewardToken	External ❗		NO ❗
└	dividendTokenBalanceOf	Public ❗		NO ❗
└	getAccountDividendsInfo	External ❗		NO ❗
└	getAccountDividendsInfoAtIndex	External ❗		NO ❗
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfDividendTokenHolders	External ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
SuperFloki_DividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		
└	setMinBalanceForDividends	External ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfTokenHolders	External ❗		NO ❗
└	getCurrentRewardToken	External ❗		NO ❗
└	getAccount	Public ❗		NO ❗
└	getAccountAtIndex	Public ❗		NO ❗
└	canAutoClaim	Private 🔐		
└	setBalance	Public ❗	🛑	onlyOwner
└	process	Public ❗	🛑	NO ❗
└	processAccount	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

POINTS TO NOTE

- Owner is not able to set buy/sell fees more than 10% (20% max fee)
- Owner is not able to set transfer fees (0% always)
- 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 is not able to mint new tokens
- **Owner must enable trading for investors**

STATIC ANALYSIS

```
Low level call in DividendPayingToken._withdrawDividendOfUser(address) (contracts/Token.sol#773-797):
        - (secondSuccess) = user.call{gas: 3000,value: _withdrawableDividend}() (contracts/Token.sol#781)
        - (success) = user.call{gas: 3000,value: _withdrawableDividend}() (contracts/Token.sol#788)
Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#953-958):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#956)
Low level call in SuperFloki.swapAndLiquify(uint256,uint256) (contracts/Token.sol#1257-1284):
        - (success) = address(dividendTracker).call{value: dividends}() (contracts/Token.sol#1281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#648) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (contracts/Token.sol#818) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/Token.sol#825) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/Token.sol#832) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/Token.sol#841) is not in mixedCase
Constant DividendPayingToken.magnitude (contracts/Token.sol#726) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter SuperFloki.setBuyTaxes(uint256,uint256,uint256)._rewards (contracts/Token.sol#1088) is not in mixedCase
Parameter SuperFloki.setBuyTaxes(uint256,uint256,uint256)._marketing (contracts/Token.sol#1088) is not in mixedCase
Parameter SuperFloki.setBuyTaxes(uint256,uint256,uint256)._liquidity (contracts/Token.sol#1088) is not in mixedCase
Parameter SuperFloki.setSellTaxes(uint256,uint256,uint256)._rewards (contracts/Token.sol#1093) is not in mixedCase
Parameter SuperFloki.setSellTaxes(uint256,uint256,uint256)._marketing (contracts/Token.sol#1093) is not in mixedCase
Parameter SuperFloki.setSellTaxes(uint256,uint256,uint256)._liquidity (contracts/Token.sol#1093) is not in mixedCase
Parameter SuperFloki.setSwapEnabled(bool)._enabled (contracts/Token.sol#1098) is not in mixedCase
Constant SuperFloki.deadWallet (contracts/Token.sol#974) is not in UPPER_CASE_WITH_UNDERSCORES
Contract SuperFloki_DividendTracker (contracts/Token.sol#1318-1524) is not in CapWords
Parameter SuperFloki_DividendTracker.getAccount(address)._account (contracts/Token.sol#1386) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#15)" inContext (contracts/Token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/Token.sol#774) is too similar to SuperFloki_DividendTracker.g
etAccount(address).withdrawableDividends (contracts/Token.sol#1393)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

SuperFloki.constructor() (contracts/Token.sol#1012-1033) uses literals with too many digits:
        - _tokengeneration(owner(),100000000 * (10 ** 9)) (contracts/Token.sol#1032)
SuperFloki.setGasForProcessing(uint256) (contracts/Token.sol#1132-1137) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,GasForProcessing must be between 200,000 and 500,000) (contracts/Token.sol#1133)
SuperFloki.slitherConstructorVariables() (contracts/Token.sol#961-1316) uses literals with too many digits:
        - gasForProcessing = 300000 (contracts/Token.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/Token.sol#516) is never used in SafeMathInt (contracts/Token.sol#514-571)
SuperFloki.currentRewardToken (contracts/Token.sol#979) is never used in SuperFloki (contracts/Token.sol#961-1316)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

SuperFloki.currentRewardToken (contracts/Token.sol#979) should be constant
SuperFloki.launchtax (contracts/Token.sol#993) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DividendPayingToken.router (contracts/Token.sol#728) should be immutable
SuperFloki.dividendTracker (contracts/Token.sol#972) should be immutable
SuperFloki.pair (contracts/Token.sol#965) should be immutable
SuperFloki.router (contracts/Token.sol#964) 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/0xb969814ff6e6abff81fc2918b955527b46002fd381abd389d0346c7bb37c0ea2

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

https://testnet.bscscan.com/tx/0x594efc4709974a50d1e7a18d36d9b0834adba2d47277bf93eb09841276a009d2

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

https://testnet.bscscan.com/tx/0xc0e144b4f6c106a5badde90f7bb2d826a2c93152d91e0643651746O5f238cbf8

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

https://testnet.bscscan.com/tx/0xde60433964f987ef7700a71292cdd07880c9add388c75267ee8a8be55b713d97

5- Buying when not excluded from fees (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0x86a6e7d4bae43766ab9185c651d9475ddbf67a57f1b1d139681a73693ef903ea

6- Selling when not excluded from fees (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0x826b572755f9ffc1c261e29d5465d5f5cdc4a86f7a9ba06933c1741256da21ae

FUNCTIONAL TESTING

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

https://testnet.bscscan.com/tx/0x7de4449a1ab0fc51dee959029841493cdf9080aa2c6a0c8f60357e084d22d3eb

8- Internal swap (passed):

As can seen in this transaction, marketing wallet received BNB

https://testnet.bscscan.com/address/0xeef5dc0425b253c321b0041ee676095e3439e30b#internaltx

9- Auto Liquidity (passed):

Auto liquidity generated tokens are burnt, as can be seen in this transaction

https://testnet.bscscan.com/token/0xacfc61a00612b33af102059e9f84f9e20f04f1f2?a=0x000000000000000000000000000000000000dead

8- Distribution of rewards (passed):

BUSD tokens are distributed between holders, this can be seen in this transaction

https://testnet.bscscan.com/tx/0x826b572755f9ffc1c261e29d5465d5f5cdc4a86f7a9ba06933c1741256da21ae

MANUAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTradingEnabled

Status: Not Resolved

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 enableTradingEnabled() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
}
```

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 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