auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

ElonFloki

DATED : 15 June 23'

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

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 enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
}
```

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 – ElonFloki

Date: 15 June, 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	1	1	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/0x94035B40df874850be42F8a8e849150CF712421F

Token Information

Token Name : Elon Floki CEO

Token Symbol: ElonFloki

Decimals: 18

Token Supply: 1,000,000,000

Token Address: ---

Checksum:
5a7c8cf5339a5b4d2ea9f1d921dfd872424e2f02

Owner:

(at time of writing the audit)

Deployer:

TOKEN OVERVIEW

Fees:

Buy Fees: 0-0.1%

Sell Fees: 0-0.1%

Transfer Fees: 0-0.1%

Fees Privilege: Owner

Ownership: owned

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - Initial distribution of the tokens

- enabling trades

- updating 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

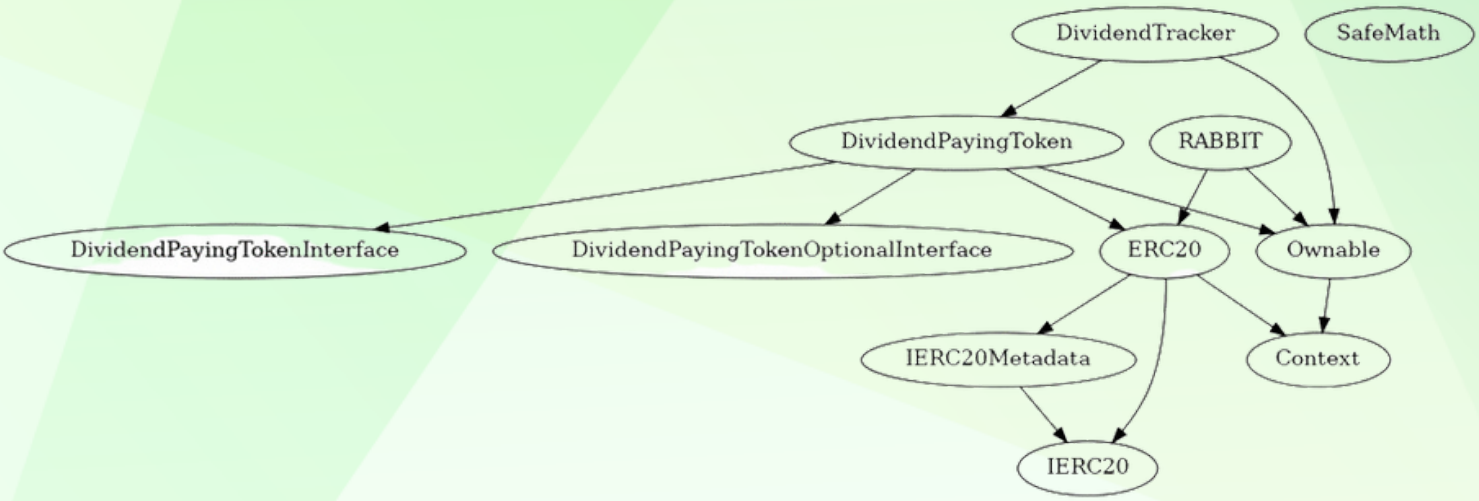
CLASSIFICATION OF RISK

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	1
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change buy/sell/transfer fees over 10% each**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- **Owner must enable trades manually**

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 ❗
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 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	

CONTRACT ASSESMENT

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 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		
IterableMapping	Library			
└	get	Internal 🔒		
└	getIndexOfKey	Internal 🔒		
└	getKeyAtIndex	Internal 🔒		
└	size	Internal 🔒		
└	set	Internal 🔒	🛑	
└	remove	Internal 🔒	🛑	
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗

CONTRACT ASSESMENT

└	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 ❗
└	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 ❗

CONTRACT ASSESMENT

└	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 ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO ❗
└	withdrawDividend	External ❗	🛑	NO ❗
DividendPayingTokenOptionalInterface	Interface			
└	withdrawableDividendOf	External ❗		NO ❗
└	withdrawnDividendOf	External ❗		NO ❗
└	accumulativeDividendOf	External ❗		NO ❗
DividendPayingToken	Implementation	ERC20, Ownable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
└	<Constructor>	Public ❗	🛑	ERC20
└	distributeDividends	Public ❗	🛑	onlyOwner
└	withdrawDividend	Public ❗	🛑	NO ❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	withdrawnDividendOf	Public ❗		NO ❗
└	accumulativeDividendOf	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	

CONTRACT ASSESMENT

└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
DividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗	|NO ❗	
└	updateMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	setLastProcessedIndex	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗	|NO ❗	
└	getNumberOfTokenHolders	External ❗	|NO ❗	
└	getAccount	Public ❗	|NO ❗	
└	getAccountAtIndex	Public ❗	|NO ❗	
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑 |NO ❗	
└	processAccount	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵 |NO ❗	
RABBIT	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	💵	ERC20
└	<Receive Ether>	External ❗	💵 |NO ❗	
└	claimStuckTokens	External ❗	🛑	onlyOwner
└	isContract	Internal 🔒		
└	sendBNB	Internal 🔒	🛑	
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	excludeFromFees	External ❗	🛑	onlyOwner
└	isExcludedFromFees	Public ❗	|NO ❗	
└	updateBuyFees	External ❗	🛑	onlyOwner
└	updateSellFees	External ❗	🛑	onlyOwner
└	changeMarketingWallet	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapAndSendDividends	Private 🔐	🛑	
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateMinimumBalanceForDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner

CONTRACT ASSESMENT

└	getClaimWait	External ❗		NO ❗
└	getTotalDividendsDistributed	External ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	dividendTokenBalanceOf	Public ❗		NO ❗
└	totalRewardsEarned	Public ❗		NO ❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	getAccountDividendsInfo	External ❗		NO ❗
└	getAccountDividendsInfoAtIndex	External ❗		NO ❗
└	processDividendTracker	External ❗	🛑	NO ❗
└	claim	External ❗	🛑	NO ❗
└	claimAddress	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO ❗
└	setLastProcessedIndex	External ❗	🛑	onlyOwner
└	getNumberOfDividendTokenHolders	External ❗		NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Pragma version^0.8.17 (contracts/Token.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in DividendPayingToken._withdrawDividendOfUser(address) (contracts/Token.sol#1211-1227):
        - (success) = user.call{gas: 50000,value: _withdrawableDividend}() (contracts/Token.sol#1217)
Low level call in RABBIT.sendBNB(address,uint256) (contracts/Token.sol#1612-1617):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#1615)
Low level call in RABBIT.swapAndSendDividends(uint256) (contracts/Token.sol#1845-1851):
        - (success) = address(address(dividendTracker)).call{value: amount}() (contracts/Token.sol#1846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#963) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#964) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#982) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/Token.sol#1002) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (contracts/Token.sol#1229) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/Token.sol#1233) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/Token.sol#1237) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/Token.sol#1241) is not in mixedCase
Constant DividendPayingToken.magnitude (contracts/Token.sol#1183) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.updateMinimumTokenBalanceForDividends(uint256)._newMinimumBalance (contracts/Token.sol#1315) is not in mixedCase
Parameter DividendTracker.getAccount(address)._account (contracts/Token.sol#1352) is not in mixedCase
Parameter RABBIT.updateBuyFees(uint256,uint256,uint256,uint256)._liquidityFeeOnBuy (contracts/Token.sol#1642) is not in mixedCase
Parameter RABBIT.updateBuyFees(uint256,uint256,uint256,uint256)._marketingFeeOnBuy (contracts/Token.sol#1643) is not in mixedCase
Parameter RABBIT.updateBuyFees(uint256,uint256,uint256,uint256)._rewardsFeeOnBuy (contracts/Token.sol#1644) is not in mixedCase
Parameter RABBIT.updateBuyFees(uint256,uint256,uint256,uint256)._trueBurnFeeOnBuy (contracts/Token.sol#1645) is not in mixedCase
Parameter RABBIT.updateSellFees(uint256,uint256,uint256,uint256)._liquidityFeeOnSell (contracts/Token.sol#1660) is not in mixedCase
Parameter RABBIT.updateSellFees(uint256,uint256,uint256,uint256)._marketingFeeOnSell (contracts/Token.sol#1661) is not in mixedCase
Parameter RABBIT.updateSellFees(uint256,uint256,uint256,uint256)._rewardsFeeOnSell (contracts/Token.sol#1662) is not in mixedCase
Parameter RABBIT.updateSellFees(uint256,uint256,uint256,uint256)._trueBurnFeeOnSell (contracts/Token.sol#1663) is not in mixedCase
Parameter RABBIT.changeMarketingWallet(address)._marketingWallet (contracts/Token.sol#1677) is not in mixedCase
Parameter RABBIT.setSwapEnabled(bool)._enabled (contracts/Token.sol#1858) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#1007) is too simil
ar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#1008)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/Token.sol#1212) is too similar to DividendTracker.getAccount(addr
ess).withdrawableDividends (contracts/Token.sol#1359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

SafeMathInt.MAX_INT256 (contracts/Token.sol#827) is never used in SafeMathInt (contracts/Token.sol#825-867)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

RABBIT.dividendTracker (contracts/Token.sol#1522) should be immutable
RABBIT.uniswapV2Pair (contracts/Token.sol#1512) should be immutable
RABBIT.uniswapV2Router (contracts/Token.sol#1511) 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

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity by owner (trades disabled) (passed):
https://testnet.bscscan.com/tx/0xc5acaa8b5f784e021d617328f50ca3d32702410d66fd6f79f9d9fe73bdbc1b18

2- Buying when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0x38379979b4dd6c5a26ee4b39ebe20229c8e5ce1b32d0e2f020a6d9830e4be4f4

3- Selling when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0x4f7f91dd91ae7862c17f59c39a60496feb2c9bb21bc2335fcc40c9d3cacfb4ab

4- Transferring when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0x0e60afc94910a60aac4a9ccc17fe1eb673d887f200733397c5f1629e5aa8f871

5- Buying when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0x4328ba6cb01e5246aa0e4534579784e237fc09c957ac1c005b0828682b3c9f9b

FUNCTIONAL TESTING

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

https://testnet.bscscan.com/tx/0x0a49b8a2b187587d501809f1bcefeeecf7e19d454fd248229016ac8eb9cb2c3b

7- Transferring when not excluded from fees (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0xd6509980b5cf5e1ffb6ea9d8532cecd6e6a7a4adcc7e66efed85aa9e038f6e49

8- Internal swap & rewards distribution (passed):

- ETH sent to marketing wallet
- A portion of tokens burnt
- A portion of tokens were added to liquidity
- A portion of tokens swapped to ETH and sent to dividend tracker
- Dividend tracker distributed reward tokens (ETH)

https://testnet.bscscan.com/tx/0x0a49b8a2b187587d501809f1bcefeeecf7e19d454fd248229016ac8eb9cb2c3b

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

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 enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
}
```

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

FUNCTIONAL TESTING

Numerics – Invlid calculation of fees

Severity: Medium

function: updateBuyFees - updateSellFees

Status: Not Resolved

Overview:

updateBuyFees and updateSellFees functions are used to update buy and sell fees. The error message of "require" statement is indicating that buy or sell fee can not exceed 10% but since 10000 is used as tax denominator, 10 is 10/10000 or only 0.1%. this means owner wont be able to set fees higher than 0.1% for buy or sells

```
function updateBuyFees(
    uint256 _liquidityFeeOnBuy,
    uint256 _marketingFeeOnBuy,
    uint256 _rewardsFeeOnBuy,
    uint256 _trueBurnFeeOnBuy
) external onlyOwner {
    liquidityFeeOnBuy = _liquidityFeeOnBuy;
    marketingFeeOnBuy = _marketingFeeOnBuy;
    rewardsFeeOnBuy = _rewardsFeeOnBuy;
    trueBurnFeeOnBuy = _trueBurnFeeOnBuy;
    totalBuyFee = liquidityFeeOnBuy + marketingFeeOnBuy + rewardsFeeOnBuy +
trueBurnFeeOnBuy;
    require(totalBuyFee <= 10, "Buy fee cannot be more than 10%");
    emit BuyFeesUpdated(totalBuyFee);
}

function updateSellFees(
    uint256 _liquidityFeeOnSell,
    uint256 _marketingFeeOnSell,
    uint256 _rewardsFeeOnSell,
    uint256 _trueBurnFeeOnSell
) external onlyOwner {
    liquidityFeeOnSell = _liquidityFeeOnSell;
    marketingFeeOnSell = _marketingFeeOnSell;
    rewardsFeeOnSell = _rewardsFeeOnSell;
    trueBurnFeeOnSell = _trueBurnFeeOnSell;
    totalSellFee = liquidityFeeOnSell + marketingFeeOnSell + rewardsFeeOnSell +
trueBurnFeeOnSell;
    require(totalSellFee <= 10, "Sell fee cannot be more than 10%");
    emit SellFeesUpdated(totalSellFee);
}
```

Suggestion

1- change "10" to "1000" in order to be abel to set buy/sell fees up to 10%

2- make sure to change "100" to "10000" in _transfer function (internal swap section) in order to have a correct tax denominator

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