AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BABYFLOKICEO

DATED : 12 APR 23'

AUDIT SUMMARY

Project name – BABYFLOKICEO

Date:12 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	1	0	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1.Manual Review:
The code has undergone a line-by-line review by the 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.

Token Information

Name : BABYFLOKICEO

Symbol : BABYFLOKICEO

Decimals: 9

Netowrk: Binance smart chain

Token Type: BEP20

Owner:
0xCbbB74c36De813C3F39347f3dD5A29323BbF26cE
(at time of audit)

Deployer:
0xFe27c8197F41acC690AF125D6577304934C91CAB

Token Address :
0x26378947D80d491D2D57878b9E94B9b3EfC5f0ac

Token Information

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 10%

Fees Privilige: owner (transfer fees)

Ownership : Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: Changing transfer fees - excluding wallets from fees - including wallets in 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

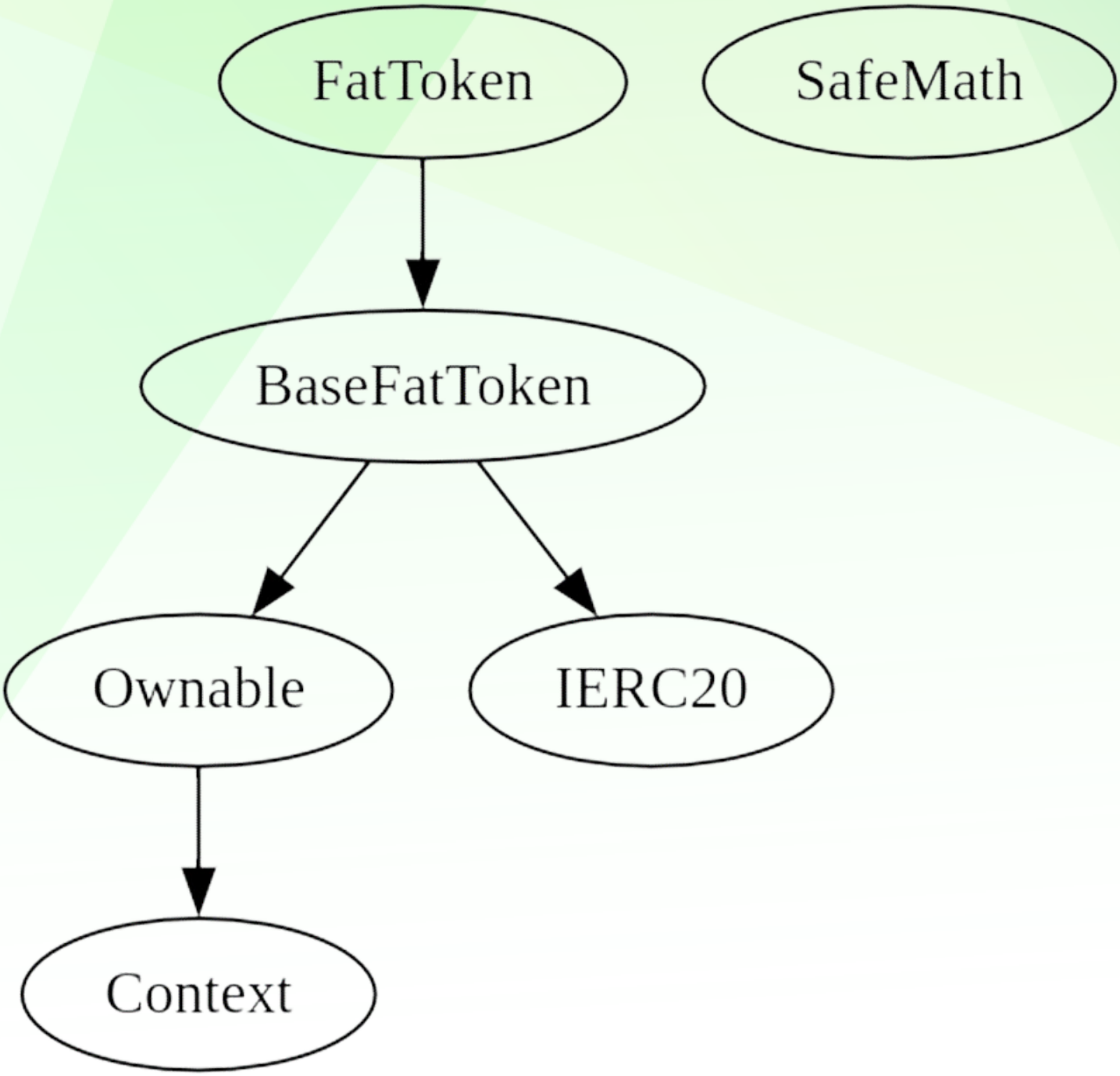
CLASSIFICATION OF RISK

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	1
Low-Risk	0
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to modify buy/sell/transfer fees (10% for each)
- 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

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	owner	Public ❗		NO ❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IERC20	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IPancakeRouter01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
IPancakeRouter02	Interface	IPancakeRouter01		

CONTRACT ASSESMENT

└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	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 ❗
BaseFatToken	Implementation	IERC20, Ownable		
└	setFundAddress	External ❗	🛑	onlyOwner
└	changeSwapLimit	External ❗	🛑	onlyOwner
└	changeWalletLimit	External ❗	🛑	onlyOwner
└	launch	External ❗	🛑	onlyOwner
└	disableSwapLimit	Public ❗	🛑	onlyOwner
└	disableWalletLimit	Public ❗	🛑	onlyOwner
└	disableChangeTax	Public ❗	🛑	onlyOwner
└	completeCustoms	External ❗	🛑	onlyOwner
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	_approve	Private 🔐	🛑	
└	setFeeWhiteList	External ❗	🛑	onlyOwner
└	multi_bclist	Public ❗	🛑	onlyOwner
TokenDistributor	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
FatToken	Implementation	BaseFatToken		
└	<Constructor>	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	setkb	Public ❗	🛑	onlyOwner
└	isReward	Public ❗		NO ❗
└	setAirDropEnable	Public ❗	🛑	onlyOwner
└	_basicTransfer	Internal 🔒	🛑	

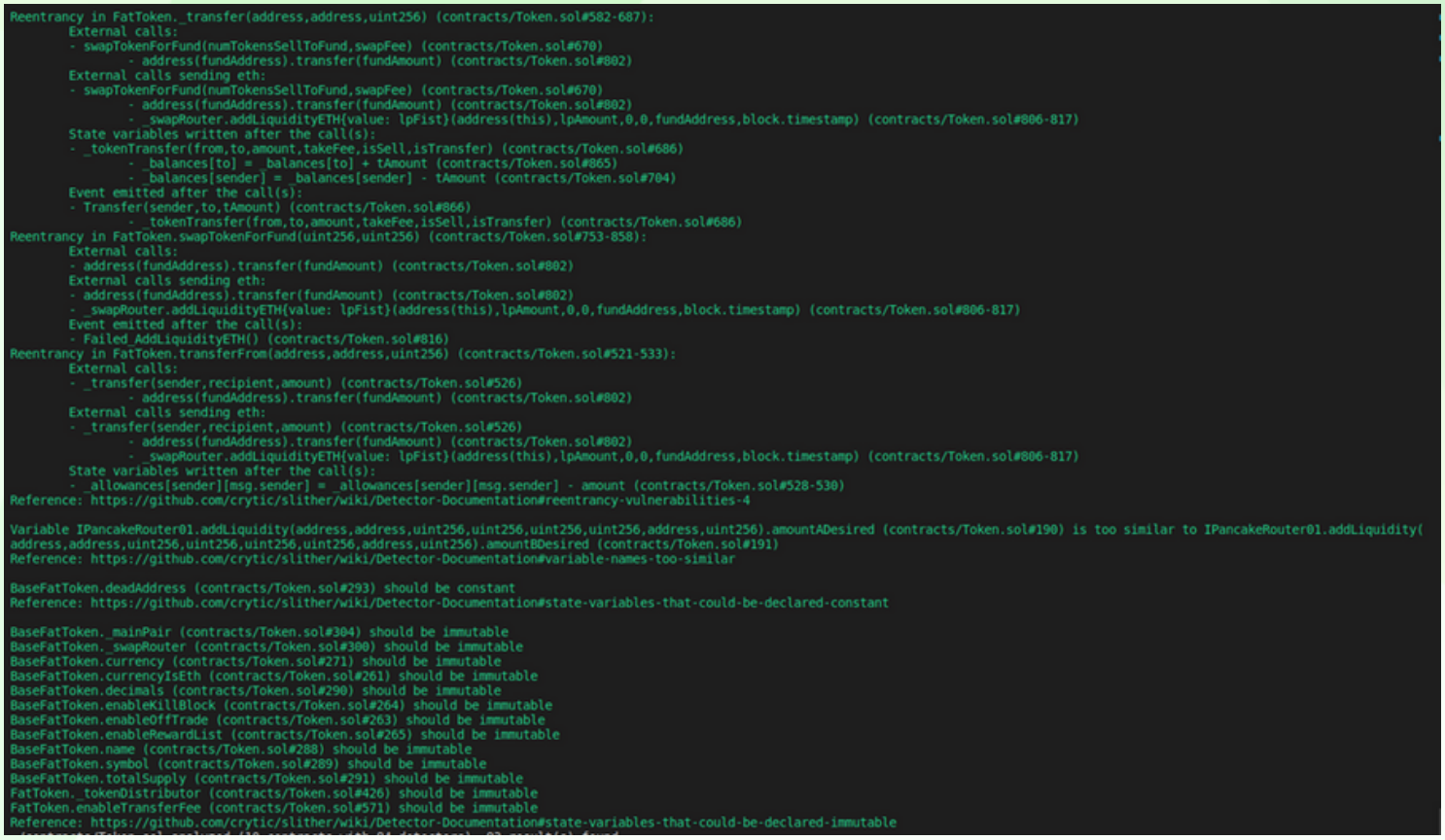
CONTRACT ASSESMENT

└	setAirdropNumbs	Public ❗	🛑	onlyOwner
└	setEnableTransferFee	Public ❗	🛑	onlyOwner
└	_transfer	Private 🔐	🛑	
└	setTransferFee	Public ❗	🛑	onlyOwner
└	_tokenTransfer	Private 🔐	🛑	
└	swapTokenForFund	Private 🔐	🛑	lockTheSwap
└	_takeTransfer	Private 🔐	🛑	
└	setSwapPairList	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reentrancy in FatToken._transfer(address,address,uint256) (contracts/Token.sol#582-687):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (contracts/Token.sol#670)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
        External calls sending eth:
        - swapTokenForFund(numTokensSellToFund,swapFee) (contracts/Token.sol#670)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
                - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#806-817)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (contracts/Token.sol#686)
                - _balances[to] = _balances[to] + tAmount (contracts/Token.sol#865)
                - _balances[sender] = _balances[sender] - tAmount (contracts/Token.sol#704)
        Event emitted after the call(s):
        - Transfer(sender,to,tAmount) (contracts/Token.sol#866)
                - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (contracts/Token.sol#686)
Reentrancy in FatToken.swapTokenForFund(uint256,uint256) (contracts/Token.sol#753-858):
        External calls:
        - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
        External calls sending eth:
        - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
        - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#806-817)
        Event emitted after the call(s):
        - Failed_AddLiquidityETH() (contracts/Token.sol#816)
Reentrancy in FatToken.transferFrom(address,address,uint256) (contracts/Token.sol#521-533):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#526)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#526)
                - address(fundAddress).transfer(fundAmount) (contracts/Token.sol#802)
                - _swapRouter.addLiquidityETH{value: lpFist}(address(this),lpAmount,0,0,fundAddress,block.timestamp) (contracts/Token.sol#806-817)
        State variables written after the call(s):
        - _allowances[sender][msg.sender] = _allowances[sender][msg.sender] - amount (contracts/Token.sol#528-530)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#190) is too similar to IPancakeRouter01.addLiquidity(
address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

BaseFatToken.deadAddress (contracts/Token.sol#293) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BaseFatToken._mainPair (contracts/Token.sol#304) should be immutable
BaseFatToken._swapRouter (contracts/Token.sol#300) should be immutable
BaseFatToken.currency (contracts/Token.sol#271) should be immutable
BaseFatToken.currencyIsEth (contracts/Token.sol#261) should be immutable
BaseFatToken.decimals (contracts/Token.sol#290) should be immutable
BaseFatToken.enableKillBlock (contracts/Token.sol#264) should be immutable
BaseFatToken.enableOffTrade (contracts/Token.sol#263) should be immutable
BaseFatToken.enableRewardList (contracts/Token.sol#265) should be immutable
BaseFatToken.name (contracts/Token.sol#288) should be immutable
BaseFatToken.symbol (contracts/Token.sol#289) should be immutable
BaseFatToken.totalSupply (contracts/Token.sol#291) should be immutable
FatToken._tokenDistributor (contracts/Token.sol#426) should be immutable
FatToken.enableTransferFee (contracts/Token.sol#571) 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 issues found

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xbfe579e7c1c1c3862c2a70b245f7cef7541614f6407441c4a2ca360b32cec4f9

2- Buying when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x12b87d7b9ba2fa039454a0b9419827bd980b25a498a1451543f485bdac35db5c

3- Selling when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xe09322663d855123b89d09a90fd48561e58db5e430e5d3f84481840c1b8bbb78

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xdf08154767c624903a08385bc5953188a96bcb6dc114590b11e522a3ed29e202

5- Buying when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0xdc6ce867240e134de031899fe477681b5f6bee8270f2739767579b63f7c3614b

6- Selling when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0x6fe056a2a14adc0703e60f28d3346c074365d929d6f142fa67f7e67c27655493

7- Transferring when not excluded from fees (up to 25% tax) (passed):
https://testnet.bscscan.com/tx/0x8ddf2524f37f7bb7438cc7e75ffef5e0095f4ff9391e61a4a2dd609876f037df

FUNCTIONAL TESTING

8- Internal swap (passed):
fee wallets received BNB
https://testnet.bscscan.com/address/0x9d54c86d4b34ef6a2dc6352235f4e0f67f501ecf#internaltx

9- Auto liquidity (passed):
Liquidity is added successfully and generated pool shares are sent to fund wallet
https://testnet.bscscan.com/token/0xc5f3a8c10e46c3e8db48d077fcee6ab599c59a9f?a=0x9d54c86d4b34ef6a2dc6352235f4e0f67f501ecf

10- Airdrop (passed):
up to 3 random wallets will receive 1 token for auto airdrop
https://testnet.bscscan.com/tx/0x6fe056a2a14adc0703e60f28d3346c074365d929d6f142fa67f7e67c27655493

MANUAL TESTING

Centralization – LP tokens sent to an EOA

Severity: Medium
Function: swapTokenForFund
Lines: 815
Status: Not Resolved
Overview:
fundAddress is receiving auto liquidity generated tokens. Overtime this tokens will be accumulated and can be used to delete a significant portion of liquidity potentially having a bad impact on token's liquidity and price

```
try
    _swapRouter.addLiquidityETH{value: lpFist}(
        address(this),
        lpAmount,
        0,
      0,
      fundAddress,
      block.timestamp
        )
{} catch {
      emit Failed_AddLiquidityETH();
}
```

Recommendation:
Introduce a minimum value safeguard for the max buy/sell/wallet amounts to prevent disabling trades. This can be achieved by setting a reasonable lower limit for both variables. For example:

MANUAL TESTING

```
try
      _swapRouter.addLiquidityETH{value: lpFist}(
          address(this),
          lpAmount,
          0,
        0,
        address(0),
        block.timestamp
          )
  {} catch {
        emit Failed_AddLiquidityETH();
  }
```

MANUAL TESTING

Gas usage – Redundant code

Severity: Informational
Function: ---
Lines: ---
Status: Not Resolved
Overview:
most sections of the code (e.g. anti-bit, max buy/sell/wallet checks, etc) are disabled forever hence are condiered redundant code
Recommendation:
its suggested to delete this sectinos of the code from contract to reduce overall gas usage and contract size

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