auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Shibarbie Token

DATED : 5 September 23'

MANUAL TESTING

Centralization – LP tokens being sent to an EOA

Severity: High
function: addLiquidity
Status: Open
Overview:
New LP tokens will be sent to owner wallet. This accumulated LP tokens can be used to remove a portion of liquidity pool.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
// approve token transfer to cover all possible scenarios
_approve(address(this), address(uniswapV2Router), tokenAmount);

// add the liquidity
uniswapV2Router.addLiquidityETH{value: ethAmount}(
address(this),
tokenAmount,
0, // slippage is unavoidable
0, // slippage is unavoidable
        owner(),
block.timestamp
);
  }
```

Suggestion
Its suggested to lock or burn new LP tokens

MANUAL TESTING

Logical – Setting swap threshold to zero

Severity: High
function: setNumTokensSellToAddToLiquidity
Status: Open
Overview:
Owner is able to set numTokensSellToAddToLiquidity to zero. If this number is set to zero, contract will try to perform swap & liquify with 0 tokens which in result, reverts the sell transaction.

```
if (
overMinTokenBalance &&
!inSwapAndLiquify &&
to == uniswapV2Pair &&
swapAndLiquifyEnabled
) {
contractTokenBalance = numTokensSellToAddToLiquidity;
//add liquidity
swapAndLiquify(contractTokenBalance);
}

function setNumTokensSellToAddToLiquidity(
uint256 newAmount
  ) external onlyOwner {
numTokensSellToAddToLiquidity = newAmount;
  }
```

Suggestion
Ensure that numTOkensSellToAddToLiquidity is always greater than a precentage of total supply

```
  function setNumTokensSellToAddToLiquidity(
uint256 newAmount
  ) external onlyOwner {
require(newAmount >= totalSupply() / 1000, “swap threshold must be greater than 0.1% of supply”);
numTokensSellToAddToLiquidity = newAmount;
  }
```

MANUAL TESTING

Logical – Division by zero

Severity: High

function: setNumTokensSellToAddToLiquidity

Status: Open

Overview:

if both _liquidityFee and _marketingFee are zero and token balance of contract is more than internal swap threshold, swapAndLiquify will fail due to a division by zero error.

```
function swapAndLiquify(uint256 contractTokenBalance) private
lockTheSwap {
uint256 tokensForLiquidity = contractTokenBalance
.mul(_liquidityFee)
.div(_liquidityFee.add(_marketingFee));
```

Suggestion

If sum of fees is zero, exit the function:

```
function swapAndLiquify(uint256 contractTokenBalance) private
lockTheSwap {
if(_liquidityFee.add(_marketingFee) == 0) return;
uint256 tokensForLiquidity = contractTokenBalance
.mul(_liquidityFee)
        .div(_liquidityFee.add(_marketingFee));
```

AUDIT SUMMARY

Project name – Shibarbie Token

Date: 5 September 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	3	0	1	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/0x160653256CeB9eC29Db166bEB42D1C8EBcd53dEE

Token Information

Token Address :
0x722dB42Bf600ceE8b560Bb789F2Cf7398f5C38eb

Name: Shibarbie Token

Symbol: SHIBARBIE

Decimals: 9

Network: Ethereum

Token Type: ERC20

Owner: 0xf7779ED8B572d7da3f8c19C2dE3dD55c60b9561e

Deployer: 0xf7779ED8B572d7da3f8c19C2dE3dD55c60b9561e

Token Supply: 10,000,000,000

Checksum:
ef746d7987bb0b51f304e4c91d7228624f8f5f8c

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/0x160653256CeB9eC29Db166bEB42D1C8EBcd53dEE

TOKEN OVERVIEW

buy fee: 0-10%

Sell fee: 0-10%

transfer fee: 0-10%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: 0.2% - 100% of total supply

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modyfing 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

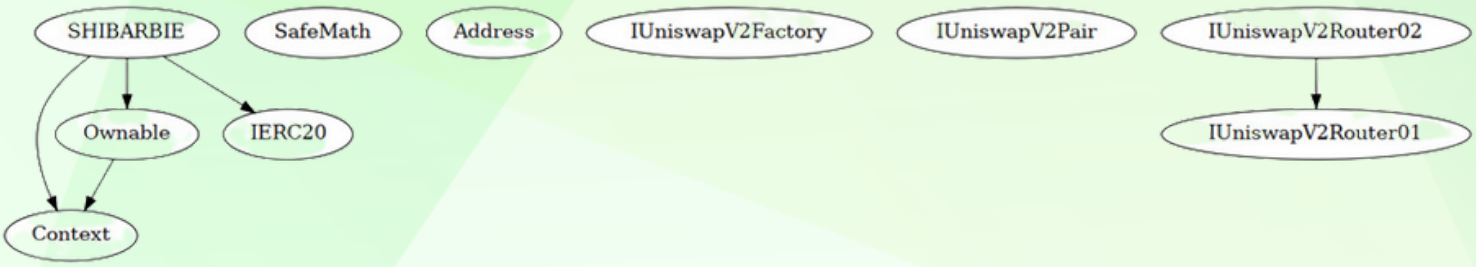
CLASSIFICATION OF RISK

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	3
Medium-Risk	0
Low-Risk	1
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to change buy/sell/transfer fees within 0-10%

- Owner is not able to blacklist an arbitrary wallet

- Owner is not able to mint new tokens

- Owner is able to set maximum buy/sell/transfer amounts (0.2% - 100% of total supply)

STATIC ANALYSIS

```
Variable SHIBARBIE.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1441) is too similar to SHIBARBIE._getTValues(uint256).tTransferAmount (c
ontracts/Token.sol#1174)
Variable SHIBARBIE.takeBurn(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1423) is too similar to SHIBARBIE.takeBurn(address,uint256,uint256,uint256).tT
ransferAmount (contracts/Token.sol#1422)
Variable SHIBARBIE._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1187) is too similar to SHIBARBIE._transferFromExcluded(address,address,uin
t256).tTransferAmount (contracts/Token.sol#1486)
Variable SHIBARBIE._getValues(uint256).rTransferAmount (contracts/Token.sol#1153) is too similar to SHIBARBIE._transferStandard(address,address,uint256).tTransferAmount (contra
cts/Token.sol#1397)
Variable SHIBARBIE.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1441) is too similar to SHIBARBIE._transferFromExcluded(address,address,u
int256).tTransferAmount (contracts/Token.sol#1486)
Variable SHIBARBIE._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1484) is too similar to SHIBARBIE._transferToExcluded(address,address,uin
t256).tTransferAmount (contracts/Token.sol#1465)
Variable SHIBARBIE._getValues(uint256).rTransferAmount (contracts/Token.sol#1153) is too similar to SHIBARBIE._getTValues(uint256).tTransferAmount (contracts/Token.sol#1174)
Variable SHIBARBIE._getValues(uint256).rTransferAmount (contracts/Token.sol#1153) is too similar to SHIBARBIE._transferFromExcluded(address,address,uint256).tTransferAmount (co
ntracts/Token.sol#1486)
Variable SHIBARBIE._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1463) is too similar to SHIBARBIE.takeBurn(address,uint256,uint256,uint256)
.tTransferAmount (contracts/Token.sol#1422)
Variable SHIBARBIE._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1118) is too similar to SHIBARBIE._getTValues(uint256).tTransferAmount (c
ontracts/Token.sol#1174)
Variable SHIBARBIE._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1118) is too similar to SHIBARBIE._transferFromExcluded(address,address,u
int256).tTransferAmount (contracts/Token.sol#1486)
Variable SHIBARBIE.takeBurn(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1423) is too similar to SHIBARBIE._getValues(uint256).tTransferAmount (contrac
ts/Token.sol#1149)
Variable SHIBARBIE.reflectionFromToken(uint256,bool).rTransferAmount (contracts/Token.sol#1073) is too similar to SHIBARBIE._transferToExcluded(address,address,uint256).tTransf
erAmount (contracts/Token.sol#1465)
Variable SHIBARBIE.takeBurn(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1423) is too similar to SHIBARBIE._transferStandard(address,address,uint256).t
TransferAmount (contracts/Token.sol#1397)
Variable SHIBARBIE._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1463) is too similar to SHIBARBIE._getValues(uint256).tTransferAmount (cont
racts/Token.sol#1149)
Variable SHIBARBIE.takeMarketing(address,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1441) is too similar to SHIBARBIE._transferToExcluded(address,address,uin
t256).tTransferAmount (contracts/Token.sol#1465)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
SHIBARBIE.slitherConstructorVariables() (contracts/Token.sol#875-1541) uses literals with too many digits:
        - _tTotal = 10000000000 * (10 ** 18) (contracts/Token.sol#889)
SHIBARBIE.slitherConstructorVariables() (contracts/Token.sol#875-1541) uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 1000000 * 10 ** 18 (contracts/Token.sol#918)
SHIBARBIE.slitherConstructorVariables() (contracts/Token.sol#875-1541) uses literals with too many digits:
        - _maxTxAmount = 500000000 * 10 ** 18 (contracts/Token.sol#919)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Loop condition `i < _excluded.length` (contracts/Token.sol#1199) should use cached array length instead of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
SHIBARBIE._decimals (contracts/Token.sol#895) should be constant
SHIBARBIE._name (contracts/Token.sol#893) should be constant
SHIBARBIE._symbol (contracts/Token.sol#894) should be constant
SHIBARBIE._tTotal (contracts/Token.sol#889) should be constant
SHIBARBIE.deadAddress (contracts/Token.sol#905) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
SHIBARBIE.uniswapV2Pair (contracts/Token.sol#913) should be immutable
SHIBARBIE.uniswapV2Router (contracts/Token.sol#912) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:./contracts/Token.sol analyzed (10 contracts with 88 detectors), 143 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	geUnlockTime	Public ❗		NO ❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑	NO ❗
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		

CONTRACT ASSESMENT

└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	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 ❗
└	DOMAIN_SEPARATOR	External ❗		NO ❗
└	PERMIT_TYPEHASH	External ❗		NO ❗
└	nonces	External ❗		NO ❗
└	permit	External ❗	🛑	NO ❗
└	MINIMUM_LIQUIDITY	External ❗		NO ❗

CONTRACT ASSESMENT

└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	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❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

CONTRACT ASSESMENT

└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
SHIBARBIE	Implementation	Context, IERC20, Ownable		
└	<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❗
└	isExcludedFromReward	Public ❗		NO❗
└	totalFees	Public ❗		NO❗
└	deliver	Public ❗	🛑	NO❗
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	_transferBothExcluded	Private 🔐	🛑	
└	<Receive Ether>	External ❗	💵	NO❗
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	removeAllFee	Private 🔐	🛑	

CONTRACT ASSESMENT

└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	takeBurn	Private 🔐	🛑	
└	takeMarketing	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setFeePercent	External ❗	🛑	onlyOwner
└	setNumTokensSellToAddToLiquidity	External ❗	🛑	onlyOwner
└	setMaxTxAmount	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x6891f2eb02a2b3c5a1a38238cccef0a14d8c35416a630a2924abe01c02f7f0a4

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5acebb24719d737d927c0e7df5ac213988fd1a158df19b5b658c5a04d535b8fc

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf0980d2879d51e67d6bb1760ba37308710ba11c0a6de4823b5a0146af4cba29d

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x2a87cf6f6c350b3fe1092ce18f8c1f3decc138ffac5ca0a947fe03c238de1e9b

5- Buying when not excluded from fees (tax 0-10%) (passed):
https://testnet.bscscan.com/tx/0x545dc498340ec49aff48cb9c1145a1ed85128a8b82a8bdabd680a85c387cc0d1

6- Selling when not excluded from fees (tax 0-10%) (passed):
https://testnet.bscscan.com/tx/0x74aa1cd16a5842d05c2143353361a31eb6ca64863ea6d2291ed99ea8dd491153

7- Transferring when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0x08dc64096894d439e39ff94b060098f243d4240ee4fef39a2e2832bbefc8184e

8- Internal swap (ETH sent to marketing wallet | Auto-liquidity)(passed):
https://testnet.bscscan.com/tx/0x74aa1cd16a5842d05c2143353361a31eb6ca64863ea6d2291ed99ea8dd491153

MANUAL TESTING

Centralization – LP tokens being sent to an EOA

Severity: High
function: addLiquidity
Status: Open
Overview:
New LP tokens will be sent to owner wallet. This accumulated LP tokens can be used to remove a portion of liquidity pool.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
// approve token transfer to cover all possible scenarios
_approve(address(this), address(uniswapV2Router), tokenAmount);

// add the liquidity
uniswapV2Router.addLiquidityETH{value: ethAmount}(
address(this),
tokenAmount,
0, // slippage is unavoidable
0, // slippage is unavoidable
        owner(),
block.timestamp
);
  }
```

Suggestion
Its suggested to lock or burn new LP tokens

MANUAL TESTING

Logical – Setting swap threshold to zero

Severity: High
function: setNumTokensSellToAddToLiquidity
Status: Open
Overview:
Owner is able to set numTokensSellToAddToLiquidity to zero. If this number is set to zero, contract will try to perform swap & liquify with 0 tokens which in result, reverts the sell transaction.

```
if (
overMinTokenBalance &&
!inSwapAndLiquify &&
to == uniswapV2Pair &&
swapAndLiquifyEnabled
) {
contractTokenBalance = numTokensSellToAddToLiquidity;
//add liquidity
swapAndLiquify(contractTokenBalance);
}

function setNumTokensSellToAddToLiquidity(
uint256 newAmount
  ) external onlyOwner {
numTokensSellToAddToLiquidity = newAmount;
  }
```

Suggestion
Ensure that numTOkensSellToAddToLiquidity is always greater than a precentage of total supply

```
  function setNumTokensSellToAddToLiquidity(
uint256 newAmount
  ) external onlyOwner {
require(newAmount >= totalSupply() / 1000, “swap threshold must be greater than 0.1% of supply”);
numTokensSellToAddToLiquidity = newAmount;
  }
```

MANUAL TESTING

Logical – Division by zero

Severity: High

function: setNumTokensSellToAddToLiquidity

Status: Open

Overview:

if both _liquidityFee and _marketingFee are zero and token balance of contract is more than internal swap threshold, swapAndLiquify will fail due to a division by zero error.

```
function swapAndLiquify(uint256 contractTokenBalance) private
lockTheSwap {
uint256 tokensForLiquidity = contractTokenBalance
.mul(_liquidityFee)
.div(_liquidityFee.add(_marketingFee));
```

Suggestion

If sum of fees is zero, exit the function:

```
function swapAndLiquify(uint256 contractTokenBalance) private
lockTheSwap {
if(_liquidityFee.add(_marketingFee) == 0) return;
uint256 tokensForLiquidity = contractTokenBalance
.mul(_liquidityFee)
        .div(_liquidityFee.add(_marketingFee));
```

MANUAL TESTING

Logical – Dead wallet receiving reflections

Severity: Low

function: takeBurn

Status: Open

Overview:

takeBurn function is not checking whether deadwallet is excluded from rewards or not. If dead wallet is excluded from rewards, _tOwned of dead wallet must be updated.

```
rTransferAmount = rTransferAmount.sub(rBurn);
tTransferAmount = tTransferAmount.sub(tBurn);
_rOwned[deadAddress] = _rOwned[deadAddress].add(rBurn);
```

Suggestion

```
rTransferAmount = rTransferAmount.sub(rBurn);
tTransferAmount = tTransferAmount.sub(tBurn);
_rOwned[deadAddress] = _rOwned[deadAddress].add(rBurn);
_tOwned[deadAddress] = _tOwned[deadAddress].add(tBurn);
```

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