AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Party Parrot

DATED : 20 May 23'

AUDIT SUMMARY

Project name – Party Parrot

Date: 20 May, 2023

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

Audit Status: **Passed with Critical Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	3	1	2	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x5f07583962f07fb052c8b44d930b615dca6713cb

Token Information

Token Name : Party Parrot

Token Symbol: PARTY

Decimals: 18

Token Supply:1,000,000,000,000

Token Address:
0xB44d8bD5346d91f0dE132ef061545B53f62344FE

Checksum:
0d99f653f7a9feb33c819908f2eac16fdd51c0f4

Owner:
0x84281c444d525A205fe511fD3D7e1F00743333B7

TOKEN OVERVIEW

Fees:

Buy Fees: 0-25%

Sell Fees: 0-25 %

Transfer Fees: 0%

Fees Privilige: owner

Ownership : owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: 0-100%

Blacklist: No

Other Priviliges: changing swap threshold - changing fees - updating router

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

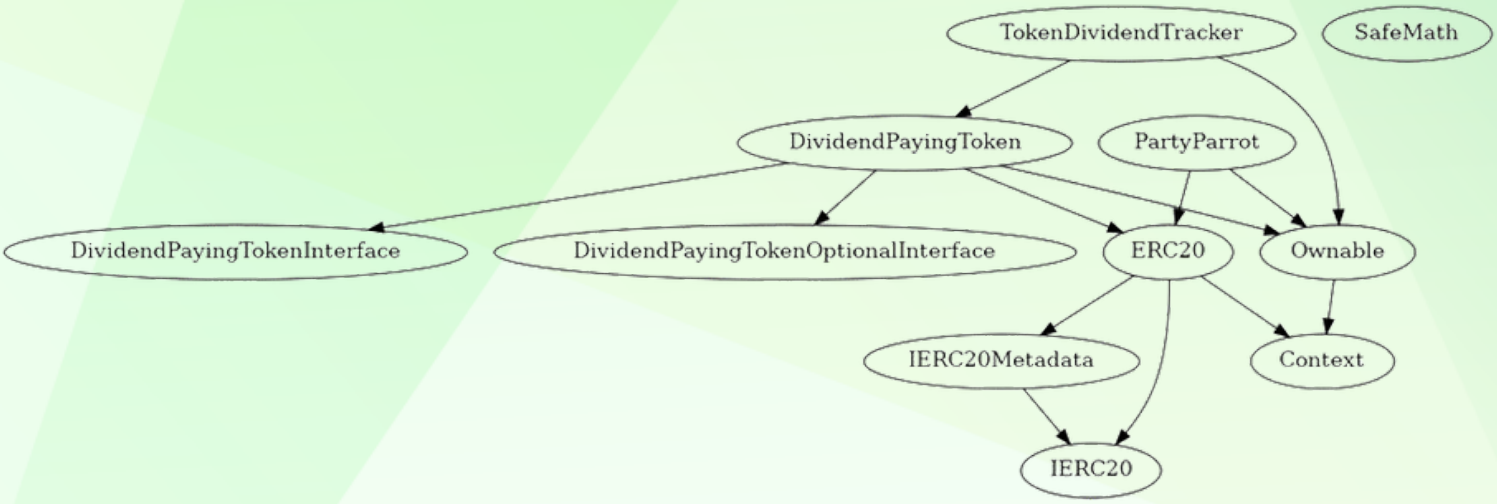
CLASSIFICATION OF RISK

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	3
High-Risk	1
Medium-Risk	2
Low-Risk	1
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change buy/sell fees each one up to 25%
- Owner is not able to blacklist an arbitrary address.
- Owner is able to disable trades but not directly
- Owner is able to burn tokens from any wallet
- Owner is not able to limit buy/sell/transfer/wallet amounts
- 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❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	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			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		

CONTRACT ASSESMENT

└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		
Clones	Library			
└	clone	Internal 🔒	🛑	
└	cloneDeterministic	Internal 🔒	🛑	
└	predictDeterministicAddress	Internal 🔒		
└	predictDeterministicAddress	Internal 🔒		
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 🔒	🛑	
└	_cast	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
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❗

CONTRACT ASSESMENT

└	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❗
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❗

CONTRACT ASSESMENT

└	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❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	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
└	distributeCAKEDividends	Public ❗	🛑	onlyOwner
└	withdrawDividend	Public ❗	🛑	NO❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	withdrawnDividendOf	Public ❗		NO❗
└	accumulativeDividendOf	Public ❗		NO❗
└	_transfer	Internal 🔒	🛑	
└	_cast	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
TokenDividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		

CONTRACT ASSESMENT

└	withdrawDividend	Public ❗			NO❗
└	setMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗			NO❗
└	getNumberOfTokenHolders	External ❗			NO❗
└	isExcludedFromDividends	Public ❗			NO❗
└	getAccount	Public ❗			NO❗
└	getAccountAtIndex	Public ❗			NO❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑		NO❗
└	processAccount	Public ❗	🛑	onlyOwner
└	MAPGet	Public ❗			NO❗
└	MAPGetIndexOfKey	Public ❗			NO❗
└	MAPGetKeyAtIndex	Public ❗			NO❗
└	MAPSize	Public ❗			NO❗
└	MAPSet	Public ❗	🛑		NO❗
└	MAPRemove	Public ❗	🛑		NO❗
PartyParrot	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	💵	ERC20
└	<Receive Ether>	External ❗	💵		NO❗
└	updateMinimumTokenBalanceForDividends	Public ❗	🛑	onlyOwner
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	burn	External ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗			NO❗
└	getTotalDividendsDistributed	External ❗			NO❗
└	isExcludedFromFees	Public ❗			NO❗
└	withdrawableDividendOf	Public ❗			NO❗
└	dividendTokenBalanceOf	Public ❗			NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	isExcludedFromDividends	Public ❗			NO❗
└	getAccountDividendsInfo	External ❗			NO❗

CONTRACT ASSESMENT

└	getAccountDividendsInfoAtIndex	External ❗			NO❗
└	processDividendTracker	External ❗	🛑		NO❗
└	claim	External ❗	🛑		NO❗
└	getLastProcessedIndex	External ❗			NO❗
└	getNumberOfDividendTokenHolders	External ❗			NO❗
└	swapManual	Public ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	Public ❗	🛑	onlyOwner
└	setDeadWallet	Public ❗	🛑	onlyOwner
└	setBuyTaxes	External ❗	🛑	onlyOwner
└	setSelTaxes	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	swapAndSendToFee	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	swapTokensForToken	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapAndSendDividends	Private 🔐	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Parameter DividendPayingToken.dividendOf(address)._owner (contracts/Token.sol#1088) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/Token.sol#1096) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/Token.sol#1105) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/Token.sol#1116) is not in mixedCase
Variable DividendPayingToken.REWARD_TOKEN (contracts/Token.sol#1004) is not in mixedCase
Constant DividendPayingToken.magnitude (contracts/Token.sol#1009) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter TokenDividendTracker.getAccount(address)._account (contracts/Token.sol#1284) is not in mixedCase
Function TokenDividendTracker.MAPGet(address) (contracts/Token.sol#1457-1459) is not in mixedCase
Function TokenDividendTracker.MAPGetIndexOfKey(address) (contracts/Token.sol#1461-1466) is not in mixedCase
Function TokenDividendTracker.MAPGetKeyAtIndex(uint256) (contracts/Token.sol#1468-1470) is not in mixedCase
Function TokenDividendTracker.MAPSize() (contracts/Token.sol#1472-1474) is not in mixedCase
Function TokenDividendTracker.MAPSet(address,uint256) (contracts/Token.sol#1476-1485) is not in mixedCase
Function TokenDividendTracker.MAPRemove(address) (contracts/Token.sol#1487-1504) is not in mixedCase
Variable PartyParrot.AmountLiquidityFee (contracts/Token.sol#1529) is not in mixedCase
Variable PartyParrot.AmountTokenRewardsFee (contracts/Token.sol#1530) is not in mixedCase
Variable PartyParrot.AmountMarketingFee (contracts/Token.sol#1531) is not in mixedCase
Variable PartyParrot._marketingWalletAddress (contracts/Token.sol#1533) is not in mixedCase
Variable PartyParrot.Optimization (contracts/Token.sol#1537) 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#632) is too simil
ar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#633)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/Token.sol#1061) is too similar to TokenDividendTracker.getAccoun
t(address).withdrawableDividends (contracts/Token.sol#1292)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Clones.clone(address) (contracts/Token.sol#238-253) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000000000000000) (contracts/Token.sol#241-244)
Clones.clone(address) (contracts/Token.sol#238-253) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000000000) (contracts/Token.sol#246-249)
Clones.cloneDeterministic(address,bytes32) (contracts/Token.sol#262-280) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000000000000000) (contracts/Token.sol#268-271
)
Clones.cloneDeterministic(address,bytes32) (contracts/Token.sol#262-280) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000000000000000) (contracts/Token.sol#
273-276)
Clones.predictDeterministicAddress(address,bytes32,address) (contracts/Token.sol#285-306) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000000000000000) (contracts/Token.so
l#292-295)
Clones.predictDeterministicAddress(address,bytes32,address) (contracts/Token.sol#285-306) uses literals with too many digits:
        - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3ff0000000000000000000000000000000000000000) (contracts/T
oken.sol#297-300)
PartyParrot.constructor(string,string,uint256,address[4],uint256[4],uint256[4],uint256) (contracts/Token.sol#1590-1658) uses literals with too many digits:
        - gasForProcessing = 300000 (contracts/Token.sol#1629)
PartyParrot.updateGasForProcessing(uint256) (contracts/Token.sol#1730-1741) 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#1731-1734)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/Token.sol#167) is never used in SafeMathInt (contracts/Token.sol#165-222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

PartyParrot.Optimization (contracts/Token.sol#1537) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DividendPayingToken.REWARD_TOKEN (contracts/Token.sol#1004) should be immutable
PartyParrot.dividendTracker (contracts/Token.sol#1515) should be immutable
PartyParrot.rewardToken (contracts/Token.sol#1517) 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

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x88ef0ba0f6bd5de063a01aa3d99a4109c77fc1ea866bea5553ebd76ce9345b1c

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x103853af66f1d5320f117e8840d02d698b7dc0d47f655f5ecf8a0474204d3836

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x9758bb0c82cb055be05ca2092a239e78e9a4bc2897e36851ed4abf409e69650c

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x2bb7f786d789f252e60deb155eb0185e2356a38302509a09678c539bdb7c5001

5- Buying when not excluded (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x9c1641cac2be90bf2031fd0c1364ab2a300fe549cac7694c2b3812e869827740

6- Selling when not excluded (0-25% tax) (failed):

FUNCTIONAL TESTING

7- Transferring when not excluded (0-25% tax) (failed):

8- Internal swap (failed):

ISSUES FOUND

Category: Centralization
Subject: Ability to burn tokens
Severity: Critical
Overview:
The contract owner has the ability to burn tokens from any other wallet without checking for allowance. This means owner can burn tokens of any other address

Code:

```
function burn(address account, uint256 amount) external onlyOwner {
    _burn(account, amount);
}
```

Suggestion:
Ensure that allowance is checked before trying to burn tokens:

```
function burn(address account, uint256 amount) external onlyOwner {
    _allowances[account][msg.sender] = _allowances[account][msg.sender] – amount;
    _burn(account, amount);
}
```

ISSUES FOUND

Category: **Centralization**
Subject: Ability to change uniswapV2Router
Severity: **Critical**
Overview:
The contract owner has the ability to update the UniswapV2Router address using the `updateUniswapV2Router` function. This could potentially allow the owner to change the router to a malicious contract which can potentially disable sell/transfers.

Code:

```
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require( newAddress != address(uniswapV2Router), "The router already has that address" );
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

Suggestion:
Ensure that uniswapV2Router is immutable and can not be changed later, this can be achieved by removing **updateUniswapV2Router** function.

ISSUES FOUND

Category: **Logical**
Subject: Invalid path for swapAndLiquify
Severity: **Critical**
Overview:
The contract owner uses WBNB as the reward token. However since the token itself is paired with WBNB a swap from token to WBNB will be failed if receiver of the WBNB tokens is one of the pair tokens (in this case, since receiver is equal to address(this) or **PARTY** token, the transaction fails with a **'UniswapV2: INVALID_TO'** Error

Code:

```
if (rewardToken == uniswapV2Router.WETH()) {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = rewardToken;
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router
        .swapExactTokensForTokensSupportingFeeOnTransferTokens(
            tokenAmount,
            0,
            path,
            address(this),
            block.timestamp
        );
```

Suggestion:
To sovle this logical issue, use swapExactTokenForETHSupportingFeeOnTransferTokens function instead of current one. This ensures that contract receives BNB instead of wrapped BNB, also its important to make furthur modifications in the contract of dividend tracker and the PARTY token. An easier solution would be to change reward token to another token except WBNB such as BUSD or USDT

ISSUES FOUND

Category: **Logical**
Subject: zero address dead wallet can disable all kind of transfers
Severity: **High**
Overview:
changing deadWallet to 0x0000000000000000000000000000000000000000 using setDeadWallet can disable all kind of transfer (buy/sell/transfer) because a transfers to 0x0000000000000000000000000000000000000000 are not allowed in the contract.

Code:
```
 function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
```

Suggestion:
Threre is not reason to add upgradeability for deadWallet, hence, ensure that deadWallet is constant and can not be updated later

ISSUES FOUND

Category: Centralization
Subject: LP tokens are sent to EOA
Severity: Medium
Overview:
LP tokens generated by swapAndLiquify are sent to an EOA

Code:

```
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
        _marketingWalletAddress,
        block.timestamp
    );
}
```

Suggestion:
its suggseted to either burn or lock new LP Tokens

ISSUES FOUND

Category: Centralization
Subject: Ability to set excessive tax fees for buy/sell/transfers
Severity: Medium
Overview:
The contract allows the owner to set tax fees for buy and sell operations. The tax fees are set using the `setBuyTaxes` and `setSelTaxes` functions, which can only be called by the contract owner. Although there is a check to ensure that the total fee does not exceed 25%, the owner can still set high fees that could negatively impact users.

Code:

```
function setBuyTaxes( uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee ) external onlyOwner {
    require( rewardsFee.add(liquidity).add(marketingFee).add(deadFee) <= 25, "Total buy fee is over 25%" );
    buyTokenRewardsFee = rewardsFee;
    buyLiquidityFee = liquidity;
    buyMarketingFee = marketingFee;
    buyDeadFee = deadFee;
}

function setSelTaxes( uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee ) external onlyOwner {
    require( rewardsFee.add(liquidity).add(marketingFee).add(deadFee) <= 25, "Total sel fee is over 25%" );
    sellTokenRewardsFee = rewardsFee;
    sellLiquidityFee = liquidity;
    sellMarketingFee = marketingFee;
    sellDeadFee = deadFee;
}
```

Suggestion:
According to pinksale safu criteria a safe range for buy/sell/transfer fees is :
buy <= 10%
sell <= 10%
transfer <= 10%

ISSUES FOUND

Category: Centralization
Subject: Control over fees for wallets
Severity: Low
Status: not applicable
Overview:
the owner can exclude wallets from fees using the `excludeFromFees` and `excludeMultipleAccountsFromFees` functions. This does not prevent the wallet from participating in buy/sell/transfers but can give the owner control over which wallets are subject to fees and which wallets are not.

Code:
```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    if (_isExcludedFromFees[account] != excluded) {
        _isExcludedFromFees[account] = excluded;
        emit ExcludeFromFees(account, excluded);
    }
}

function excludeMultipleAccountsFromFees( address[] calldata accounts, bool excluded ) public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }
    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

Suggestion:
This issue has a low severity, as it does not directly impact the ability to buy/sell/transfer tokens. However, to further reduce centralization, consider implementing a mechanism that allows the community to vote on which wallets should be excluded from fees.

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