auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PepeVault

DATED : 11 May 23'

AUDIT SUMMARY

Project name – PepeVault

Date: 11 May, 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	0	1	2
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/0xDFeB2c192CF9a05690f4f0C4443Fec5ee8730184

Token Information

Token Name : PepeVault

Token Symbol: PEPEV

Decimals: 18

Token Supply: 100,000,000,000

Token Address:
0x43591f7933Ee1CF68e8fbae6dDaB06B0EC3696e5

Checksum:
54244f4f6cf5eda35b0b3827c0ca73fe7afe3a19

Owner:
0xe39eb5113c106f7dfc7d49b175466e82bf88bdcd

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: changing fees - changing 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

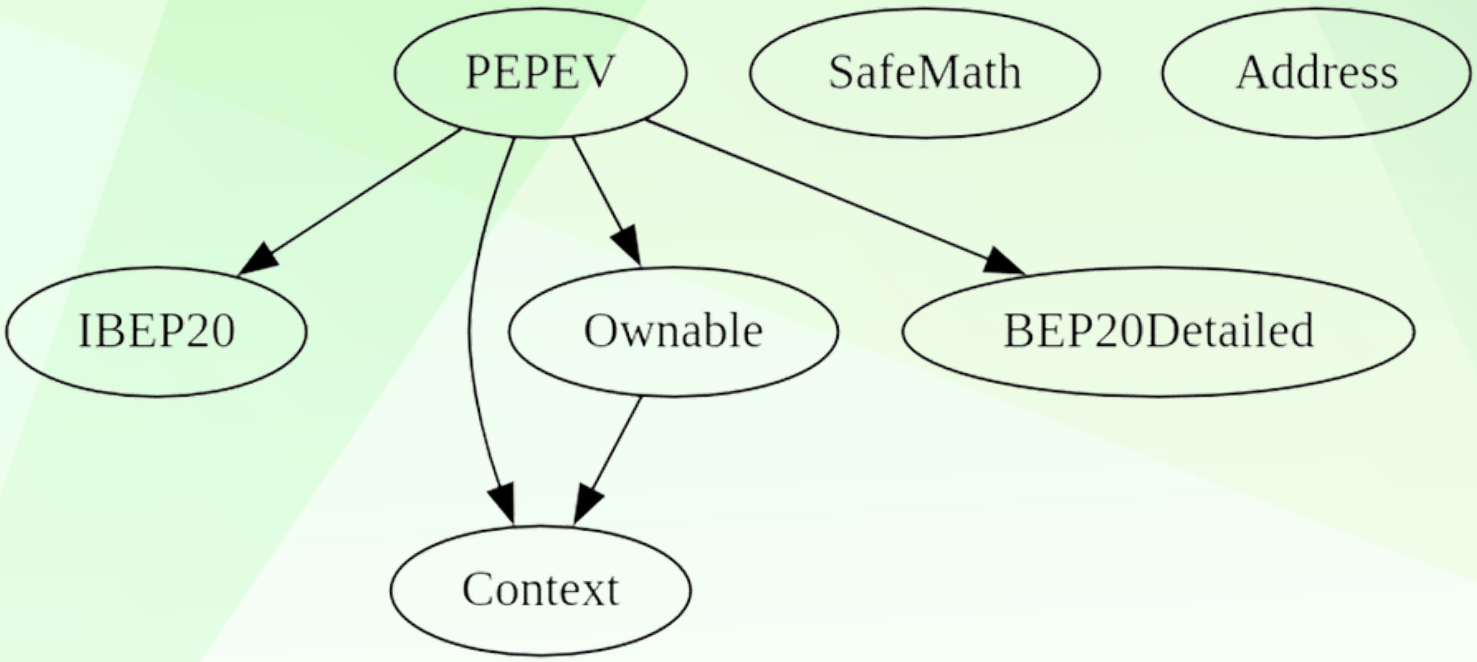
CLASSIFICATION OF RISK

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	0
Low-Risk	1
Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set sell/buy tax higher than 10% each
- Owner is not able to set transfer tax (0%)
- Owner is not able to set a max buy/transfer/wallet/sell 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
- Trades are enabled, no need for manual enabling

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	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 🔒		
Context	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	_msgSender	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
BEP20Detailed	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
Address	Library			

CONTRACT ASSESMENT

└	isContract	Internal 🔒		
SafeBEP20	Library				
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeApprove	Internal 🔒	🛑	
└	callOptionalReturn	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 ❗	
└	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 ❗	

CONTRACT ASSESMENT

└	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 ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
PEPEV	Implementation	Context, Ownable, IBEP20, BEP20Detailed		
└	<Constructor>	Public ❗	🛑	BEP20Detailed
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗

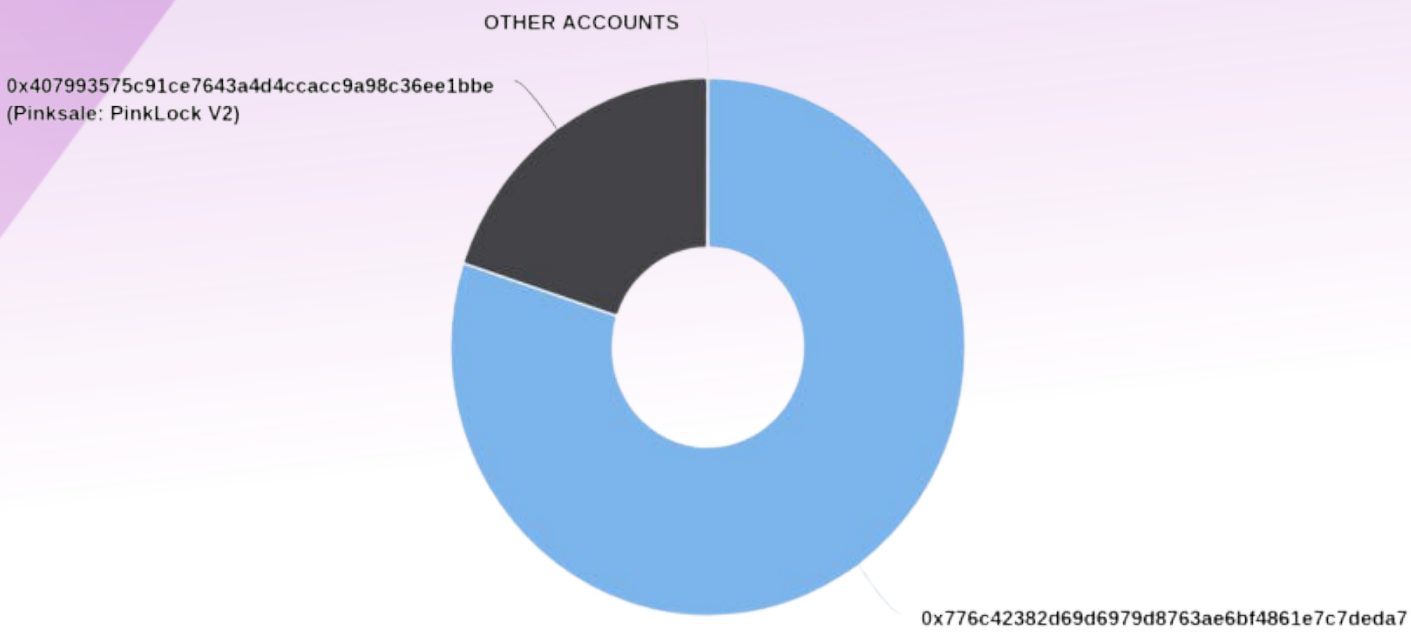
CONTRACT ASSESMENT

└	decreaseAllowance	Public ❗	🛑	NO ❗
└	setBuyMarketingFeePercent	External ❗	🛑	onlyOwner
└	setSellMarketingFeePercent	External ❗	🛑	onlyOwner
└	setMarketingAddress	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	changeNumTokensSellToFee	External ❗	🛑	onlyOwner
└	clearBNB	External ❗	🛑	onlyOwner
└	clearBEP20	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	_approve	Internal 🔒	🛑	

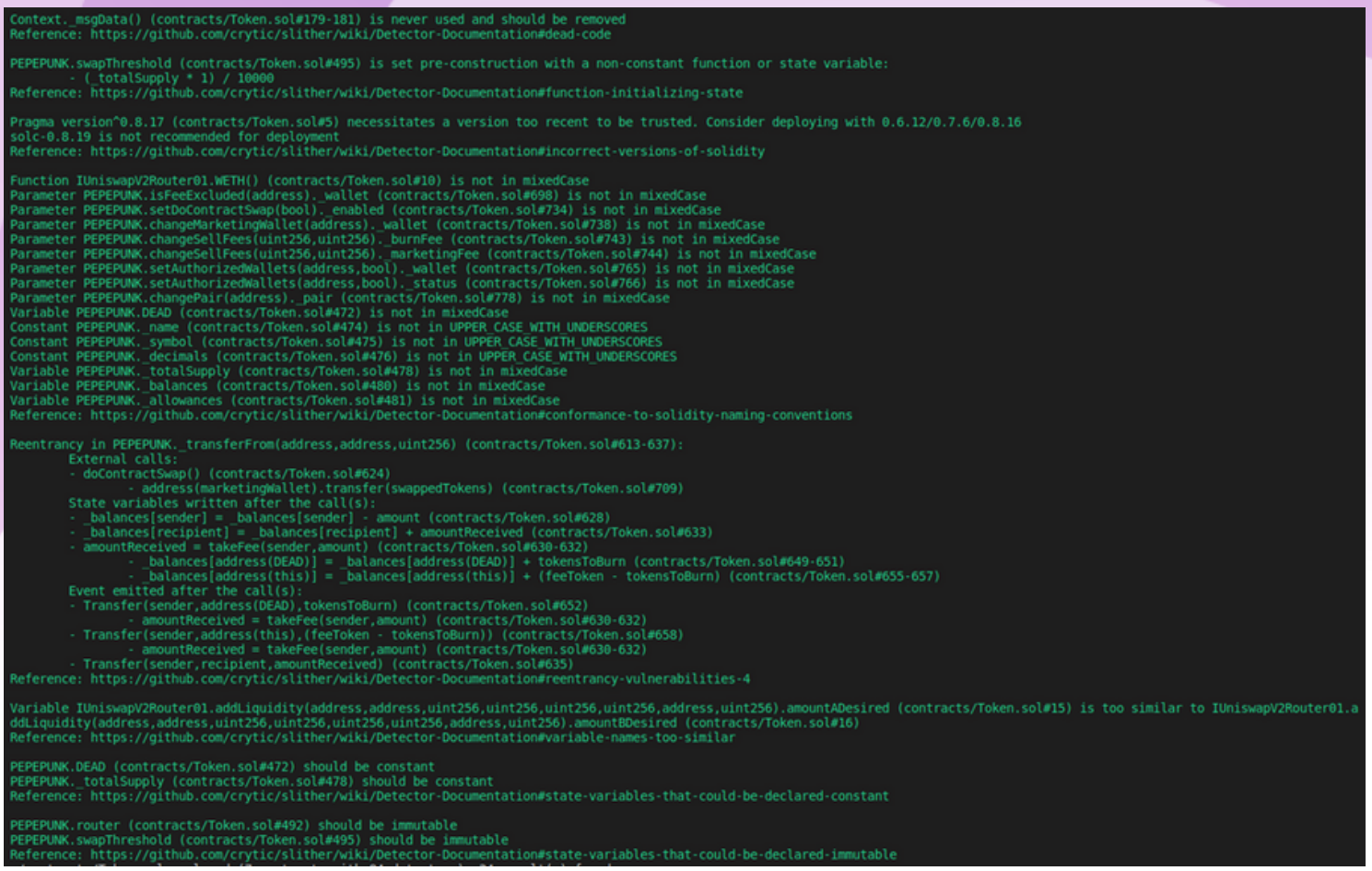
Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

TOKENOMICS AT TIME OF AUDIT

Holders at time of audit



STATIC ANALYSIS



```
Context._msgData() (contracts/Token.sol#179-181) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

PEPEPUNK.swapThreshold (contracts/Token.sol#495) is set pre-construction with a non-constant function or state variable:
        - (_totalSupply * 1) / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (contracts/Token.sol#10) is not in mixedCase
Parameter PEPEPUNK.isFeeExcluded(address)._wallet (contracts/Token.sol#698) is not in mixedCase
Parameter PEPEPUNK.setDoContractSwap(bool)._enabled (contracts/Token.sol#734) is not in mixedCase
Parameter PEPEPUNK.changeMarketingWallet(address)._wallet (contracts/Token.sol#738) is not in mixedCase
Parameter PEPEPUNK.changeSellFees(uint256,uint256)._burnFee (contracts/Token.sol#743) is not in mixedCase
Parameter PEPEPUNK.changeSellFees(uint256,uint256)._marketingFee (contracts/Token.sol#744) is not in mixedCase
Parameter PEPEPUNK.setAuthorizedWallets(address,bool)._wallet (contracts/Token.sol#765) is not in mixedCase
Parameter PEPEPUNK.setAuthorizedWallets(address,bool)._status (contracts/Token.sol#766) is not in mixedCase
Parameter PEPEPUNK.changePair(address)._pair (contracts/Token.sol#778) is not in mixedCase
Variable PEPEPUNK.DEAD (contracts/Token.sol#472) is not in mixedCase
Constant PEPEPUNK._name (contracts/Token.sol#474) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPEPUNK._symbol (contracts/Token.sol#475) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPEPUNK._decimals (contracts/Token.sol#476) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PEPEPUNK._totalSupply (contracts/Token.sol#478) is not in mixedCase
Variable PEPEPUNK._balances (contracts/Token.sol#480) is not in mixedCase
Variable PEPEPUNK._allowances (contracts/Token.sol#481) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in PEPEPUNK._transferFrom(address,address,uint256) (contracts/Token.sol#613-637):
        External calls:
        - doContractSwap() (contracts/Token.sol#624)
                - address(marketingWallet).transfer(swappedTokens) (contracts/Token.sol#709)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender] - amount (contracts/Token.sol#628)
        - _balances[recipient] = _balances[recipient] + amountReceived (contracts/Token.sol#633)
        - amountReceived = takeFee(sender,amount) (contracts/Token.sol#630-632)
                - _balances[address(DEAD)] = _balances[address(DEAD)] + tokensToBurn (contracts/Token.sol#649-651)
                - _balances[address(this)] = _balances[address(this)] + (feeToken - tokensToBurn) (contracts/Token.sol#655-657)
        Event emitted after the call(s):
        - Transfer(sender,address(DEAD),tokensToBurn) (contracts/Token.sol#652)
                - amountReceived = takeFee(sender,amount) (contracts/Token.sol#630-632)
        - Transfer(sender,address(this),(feeToken - tokensToBurn)) (contracts/Token.sol#658)
                - amountReceived = takeFee(sender,amount) (contracts/Token.sol#630-632)
        - Transfer(sender,recipient,amountReceived) (contracts/Token.sol#635)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#15) is too similar to IUniswapV2Router01.a
ddLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PEPEPUNK.DEAD (contracts/Token.sol#472) should be constant
PEPEPUNK._totalSupply (contracts/Token.sol#478) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PEPEPUNK.router (contracts/Token.sol#492) should be immutable
PEPEPUNK.swapThreshold (contracts/Token.sol#495) 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/0x5cd86c9f00f2d411c2fd54bc74806c4ccc158771a1bb246504e1c6912c00169c

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

https://testnet.bscscan.com/tx/0x46afa057efaad1574acf0cf132e2300e57e96af204da99e107df2d4a14377ed0

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

https://testnet.bscscan.com/tx/0x432805eef9610ff36da7834198e62d28f58f3379ca0a85a0b592bc5d3a55ac94

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

https://testnet.bscscan.com/tx/0xc9132894e1f436fc0955714e9f0be3de9bdd3647842930df912a38baa0bf5e81

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

https://testnet.bscscan.com/tx/0x1d46ca7837dc0355bc2280703e4b0fcd05b8cca00d7b5181a3e7ca755c1eb976

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

https://testnet.bscscan.com/tx/0xcb1dbc02365fc6904418e3678b074bb01680a4b9abb10c6e79f2aadcd340e415

FUNCTIONAL TESTING

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

https://testnet.bscscan.com/tx/0x3efe2a34de92a05105b6d30bdde527ba85b3fe7917496d5a3470af760e7a3f98

7- Internal swap (marketing wallet received BNB) (passed):

https://testnet.bscscan.com/address/0x8aeb884284e284d7c598523e991ca9ccd3b0f66a#internaltx

MANUAL TESTING

Logical – Marketing wallet

Status: Not Resolved
Severity: Low
Overview:
Marketing wallet can be set to a contract. If contract rejects receiving BNB the internal swap would be failed.

```
Contract marketingWallet {
        receive() external payable {revert();}
}
```

Suggestion:
ensure that marketing wallet is not a contract.

MANUAL TESTING

Informational – Withdrawal of native tokens

Status: Not Resolved

Overview:

Owner of the contract is able to withdraw native tokens from the contract.

Suggestion:

implement a function to be able to withraw ERC20 tokens from the contract

MANUAL TESTING

Informational – using “transfer” for sending BNB

Status: Not Resolved

Overview:

contract is using “transfer” function to transfer BNB to marketing wallet. This function forwards a fixed amount of gas, if marketing wallet executes code in its receive function, there wont be enough gas for it.

```
Contract MarketingWallet {
        receive() external payable{
                //performs some code here but no gas
        }
}
```

Suggestion:

implement a function to be able to withraw ERC20 tokens from the contract

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