



# Smart Contract Audit

FOR  
**PEPE AI CEO**

DATED : 8 May 23'



# AUDIT SUMMARY

**Project name - PEPE AI CEO**

**Date:** 8 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	0	13
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/0x6837f492645276d0c9be69c41a40a63ea5db1bf6>

---



# Token Information

---

**Token Name :** PEPE AI CEO

**Token Symbol:** PEPE

**Decimals:** 9

**Token Supply:** 100,000,000,000,000

**Token Address:**

0x36c0a3Ed58a6102001FdA8cE88F8fC707ce242aD

**Checksum:**

51b0cd22e4a4e063910624f3874a98f24bcebb2b

**Owner:**

0xdd0253F0D31369a693836A26E545fbD66E0C97fE

**(at time of writing the audit)**

**Deployer:**

0x0Bd22D8a2624A16847aF7aedAB6ddCd9cF84c50b

---



# TOKEN OVERVIEW

---

**Fees:**

Buy Fees: 5%

Sell Fees: 5%

Transfer Fees: 5%

---

**Fees Privilege:** Owner

---

**Ownership:** Owned

---

**Minting:** No mint function

---

**Max Tx Amount/ Max Wallet Amount:** No

---

**Blacklist:** No

---

**Other Privileges:** changing swap threshold

---





# 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 is exercised 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



**Gasless Send**



Private modifier



Using block.timestamp



Multiple Sends



Re-entrancy



Using Suicide



Tautology or contradiction



Gas Limit and Loops



Timestamp Dependence



Address hardcoded



Revert/require functions



Exception Disorder



Use of tx.origin



Using inline assembly



Integer overflow/underflow



Divide before multiply



Dangerous strict equalities



Missing Zero Address Validation



Using SHA3



Compiler version not fixed



Using throw

# CLASSIFICATION OF RISK

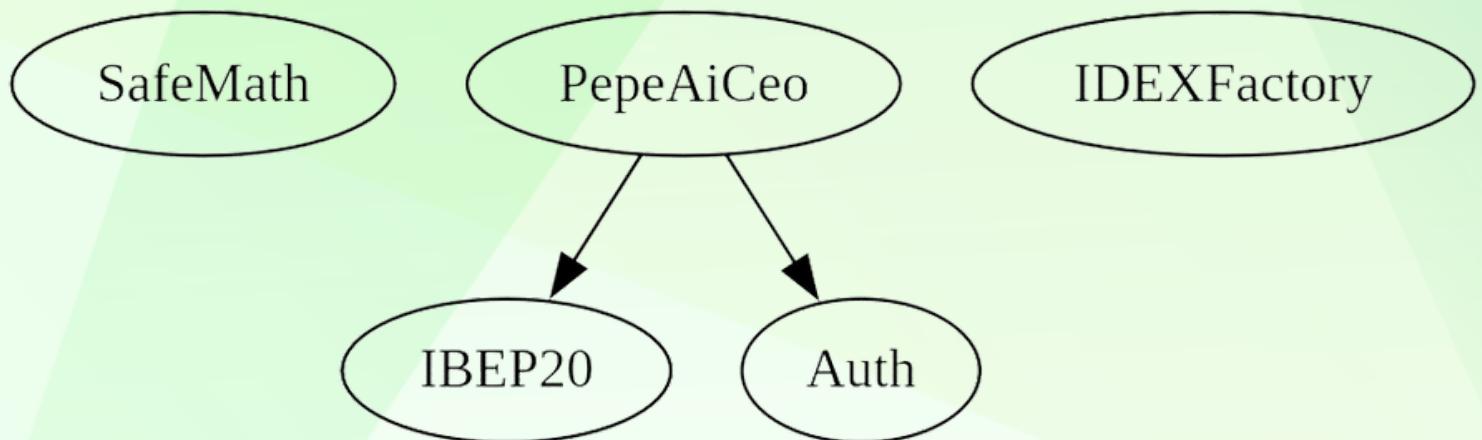
Severity	Description
◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a 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	0
◆ Gas Optimization / Suggestions	13

# INHERITANCE TREE

---





## POINTS TO NOTE

---

It should be noted that below details only apply to the current version of the contract deployed at the address :

**0x36c0a3Ed58a6102001FdA8cE88F8fC707ce242aD**

- Owner is not able to change buy/sell/transfer fees at current version of the contract (5%)
- 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 able to limit buys/transfers/sells by a max amount as limit
- Owner is not able to mint new tokens
- Trades are already enable



# CONTRACT ASSESSMENT

Contract	Type	Bases			
	L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
**IERC165**	Interface				
L	supportsInterface	External !	NO !		
**SafeMath**	Library				
L	tryAdd	Internal			
L	trySub	Internal			
L	tryMul	Internal			
L	tryDiv	Internal			
L	tryMod	Internal			
L	add	Internal			
L	sub	Internal			
L	mul	Internal			
L	div	Internal			
L	mod	Internal			
L	sub	Internal			
L	div	Internal			
L	mod	Internal			
**IBEP20**	Interface				
L	totalSupply	External !	NO !		
L	decimals	External !	NO !		
L	symbol	External !	NO !		
L	name	External !	NO !		
L	getOwner	External !	NO !		
L	balanceOf	External !	NO !		
L	transfer	External !		NO !	
L	burn	External !		NO !	
L	allowance	External !	NO !		
L	approve	External !		NO !	
L	transferFrom	External !		NO !	
**Auth**	Implementation				
L	<Constructor>	Public !		NO !	
L	isOwner	Public !	NO !		
L	renounceOwnership	Public !		onlyOwner	
L	transferOwnership	Public !		onlyOwner	
**IDEXFactory**	Interface				
L	createPair	External !		NO !	



# CONTRACT ASSESSMENT

	**IDEXRouter**   Interface	
L	factory   External !   NO !	
L	WETH   External !   NO !	
L	addLiquidity   External !   ●   NO !	
L	addLiquidityETH   External !   \$   NO !	
L	swapExactTokensForTokensSupportingFeeOnTransferTokens   External !   ●   NO !	
L	swapExactETHForTokensSupportingFeeOnTransferTokens   External !   \$   NO !	
L	swapExactTokensForETHSupportingFeeOnTransferTokens   External !   ●   NO !	
	**IDividendDistributor**   Interface	
L	setDistributionCriteria   External !   ●   NO !	
L	setShare   External !   ●   NO !	
L	deposit   External !   \$   NO !	
L	process   External !   ●   NO !	
	**DividendDistributor**   Implementation   IDividendDistributor	
L	<Constructor>   Public !   ●   NO !	
L	setDistributionCriteria   External !   ●   onlyToken	
L	setShare   External !   ●   onlyToken	
L	deposit   External !   \$   onlyToken	
L	process   External !   ●   onlyToken	
L	shouldDistribute   Internal 🔒	
L	distributeDividend   Internal 🔒   ●	
L	claimDividend   External !   ●   onlyToken	
L	getUnpaidEarnings   Public !     NO !	
L	getCumulativeDividends   Internal 🔒	
L	addShareholder   Internal 🔒   ●	
L	removeShareholder   Internal 🔒   ●	
L	setDividendTokenAddress   External !   ●   onlyToken	
	**PepeAiCeo**   Implementation   IBEP20, Auth	
L	<Constructor>   Public !   ●   Auth	
L	<Receive Ether>   External !   \$   NO !	
L	totalSupply   External !     NO !	
L	decimals   External !     NO !	
L	symbol   External !     NO !	
L	name   External !     NO !	
L	getOwner   External !     NO !	
L	balanceOf   Public !     NO !	
L	allowance   External !     NO !	
L	approve   Public !   ●   NO !	
L	approveMax   External !   ●   NO !	



# CONTRACT ASSESSMENT

L   burn   External !	●   NO !
L   transfer   External !	●   NO !
L   transferFrom   External !	●   NO !
L   _transferFrom   Internal 🔒	●
L   _basicTransfer   Internal 🔒	●
L   shouldTakeFee   Internal 🔒	
L   shouldTakeFee   Internal 🔒	
L   getTotalFee   Public !	NO !
L   getMultipliedFee   Public !	NO !
L   takeFee   Internal 🔒	●
L   shouldSwapBack   Internal 🔒	
L   swapBack   Internal 🔒	●   swapping
L   buyTokens   Internal 🔒	●   swapping
L   launched   Internal 🔒	
L   setIsDividendExempt   External !	●   onlyOwner
L   setIsFeeExempt   External !	●   onlyOwner
L   setFeeReceivers   External !	●   onlyOwner
L   setSwapBackSettings   External !	●   onlyOwner
L   setTargetLiquidity   External !	●   onlyOwner
L   manualSend   External !	●   NO !
L   setDistributionCriteria   External !	●   onlyOwner
L   claimDividend   External !	●   NO !
L   getUnpaidEarnings   Public !	NO !
L   setDistributorSettings   External !	●   onlyOwner
L   getCirculatingSupply   Public !	NO !
L   getLiquidityBacking   Public !	NO !
L   isOverLiquified   Public !	NO !

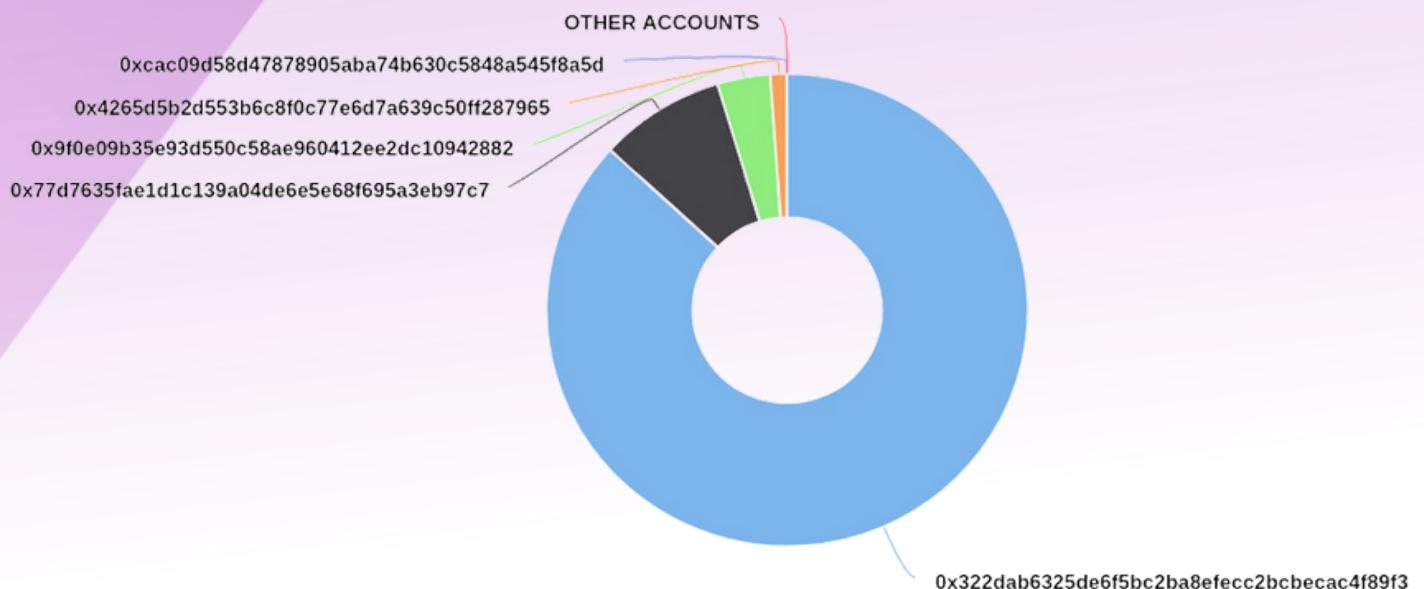
## Legend

Symbol	Meaning
●	Function can modify state
💵	Function is payable

# TOKENOMICS AT TIME OF AUDIT

PEPE AI CEO Top 100 Token Holders

Source: BscScan.com





# STATIC ANALYSIS

```
Variable PepeAiCeo.REWARD (contracts/Token.sol#469) is not in mixedCase
Variable PepeAiCeo.WBNB (contracts/Token.sol#470) is not in mixedCase
Variable PepeAiCeo.DEAD (contracts/Token.sol#471) is not in mixedCase
Variable PepeAiCeo.ZERO (contracts/Token.sol#472) is not in mixedCase
Constant PepeAiCeo.name (contracts/Token.sol#474) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepeAiCeo.symbol (contracts/Token.sol#475) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepeAiCeo.decimals (contracts/Token.sol#476) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PepeAiCeo.totalSupply (contracts/Token.sol#477) is not in mixedCase
Variable PepeAiCeo.balances (contracts/Token.sol#479) is not in mixedCase
Variable PepeAiCeo.allowances (contracts/Token.sol#480) is not in mixedCase
Variable PepeAiCeo.ts.Project (contracts/Token.sol#510) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#220) is too similar to IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PepeAiCeo.slitherConstructorVariables() (contracts/Token.sol#466-929) uses literals with too many digits:
  - _totalsupply = 1000000000000000 * (10 ** _decimals) (contracts/Token.sol#477)
PepeAiCeo.slitherConstructorVariables() (contracts/Token.sol#466-929) uses literals with too many digits:
  - distributorGas = 300000 (contracts/Token.sol#515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PepeAiCeo.REWARD (contracts/Token.sol#469) is never used in PepeAiCeo (contracts/Token.sol#466-929)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DividendDistributor.WBNB (contracts/Token.sol#289) should be constant
DividendDistributor.dividendsPerShareAccuracyFactor (contracts/Token.sol#299) should be constant
PepeAiCeo.DEAD (contracts/Token.sol#471) should be constant
PepeAiCeo.REWARD (contracts/Token.sol#469) should be constant
PepeAiCeo.WBNB (contracts/Token.sol#470) should be constant
PepeAiCeo.ZERO (contracts/Token.sol#472) should be constant
PepeAiCeo._totalSupply (contracts/Token.sol#477) should be constant
PepeAiCeo.buyFeeDenominator (contracts/Token.sol#489) should be constant
PepeAiCeo.buyFeeBurn (contracts/Token.sol#490) should be constant
PepeAiCeo.liquidityBuyFee (contracts/Token.sol#485) should be constant
PepeAiCeo.liquiditySellFee (contracts/Token.sol#492) should be constant
PepeAiCeo.marketingBuyFee (contracts/Token.sol#486) should be constant
PepeAiCeo.marketingSellFee (contracts/Token.sol#493) should be constant
PepeAiCeo.projectBuyFee (contracts/Token.sol#487) should be constant
PepeAiCeo.projectSellFee (contracts/Token.sol#494) should be constant
PepeAiCeo.reflectionBuyFee (contracts/Token.sol#503) should be constant
PepeAiCeo.reflectionSellFee (contracts/Token.sol#504) should be constant
PepeAiCeo.sellFeeDenominator (contracts/Token.sol#496) should be constant
PepeAiCeo.sellFeeBurn (contracts/Token.sol#497) should be constant
PepeAiCeo.totalBuyFee (contracts/Token.sol#488) should be constant
PepeAiCeo.totalSellFee (contracts/Token.sol#495) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DividendDistributor._token (contracts/Token.sol#280) should be immutable
DividendDistributor.router (contracts/Token.sol#290) should be immutable
PepeAiCeo.distributor (contracts/Token.sol#514) should be immutable
PepeAiCeo.launchedAt (contracts/Token.sol#512) should be immutable
PepeAiCeo.launchedAtTimestamp (contracts/Token.sol#513) should be immutable
PepeAiCeo.pair (contracts/Token.sol#509) should be immutable
PepeAiCeo.router (contracts/Token.sol#508) should be immutable
PepeAiCeo.ts.Project (contracts/Token.sol#510) 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/0xf4bead34d6a1a5332ec2aefc0a2002f4f05d97ec48e245be53eabfec8f4053c0>

### 2- Buying when excluded from fees (0% tax) (**passed**):

<https://testnet.bscscan.com/tx/0x53aa2ce7521c65ade64109cca4d9ef6e97d9e68bf957b9dd78963223158dbe94>

### 3- Selling when excluded from fees (0% tax) (**passed**):

<https://testnet.bscscan.com/tx/0xe2226e2b6f8e824f1df4eb15398daff1e351826f780fe33290e58d63887f5afd>

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

<https://testnet.bscscan.com/tx/0xb1b134387dd311d55d51d185c0a6a93b56dab4b0dec0ff68dc8582f9440afada>

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

<https://testnet.bscscan.com/tx/0x681968e9746c504523e3ed61cd3d3327f8126ba61e49a68daed9d6dd28900cca>

### 6- Selling when not excluded from fees ( 5% tax ) (**passed**):

<https://testnet.bscscan.com/tx/0x149a068012cb50fc193359c0c562eed6440cef1f75d349fbb3c03f4201b664e1>

---



# FUNCTIONAL TESTING

---

## 7- Transferring when not excluded from fees ( 5% tax ) (passed):

<https://testnet.bscscan.com/tx/0x46a61f3ceabe43a46bf6f33356094d85dc0be6f8eddf004c76bea6bdf364605c>

## 8- Internal swap (passed):

fee wallets received BNB

<https://testnet.bscscan.com/address/0xbFD4527f574c755F94d1341b5010536355B464f7#internaltx>



# MANUAL TESTING

## Suggestions

- Change name of **savetokens** function to something that identifies its action, like “withdrawTokens”
- There are two functions that are doing exact same thing. One of this functions can be removed; **shouldTakeFee**, **shouldTakeFee**
- Redundant function : **buyTokens**  
Event if burn and contract are not receiving any tokens (from tax) a Transfer event is still emitted, this may cause confusion for investors
  - **emit Transfer(sender, address(this), feeamount2);**
  - **emit Transfer(sender, DEAD, amounttoburn);**
- Some features are disabled in the contract, like auto-liquidity – burns and rewards, having redundant code related to this features only increases gas usage with no purpose  
**Fees can not be changed, consider adding a function to be able to change fees within a safe range**

## Gas optimizations

- Define **router** variable as constant
- Define **pair** variable as constant
- Define **REWARD**, **WBNB**, **DEAD**, **ZERO** as constant variables
- Redundant code here:  
**\_balances[Project] = (\_totalSupply \* 100) / 100;**
- Can be changed to  
**\_balances[Project] = \_totalSupply;**  
Redundant code at **\_transfer** function  
**if (sender != pair && !isOwner(sender)) {}**
- Redundant code at **\_transfer** function, since **emergBlock** is always set to true  
**if (!authorizations[sender] && !authorizations[recipient]) {  
    require(emergBlock, "Trading not open yet");  
}**



# 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 specialize 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://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---