auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Pepe Gangster

DATED : 14 May 23'

AUDIT SUMMARY

Project name – Pepe Gangster

Date: 14 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	1	0	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/tx/0x634f6cbeb1c5e3b29869628385616cf622670cf1223aa69720a49ea41f89aa70

Token Information

Token Name : Pepe Gangster

Token Symbol: PEPEG

Decimals: 9

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

Token Address:
0x8481c3Da0b3084664050acd956a2081dA3Aae5A0

Checksum:
b5bb614de2020c290c8d9c47740f00045d91de46

Owner:
0xBbCe79A675324713d6F2A933BC7af22e5D9925d2

TOKEN OVERVIEW

Fees:

Buy Fees: upto 5%

Sell Fees: upto 5 %

Transfer Fees: upto 5%

Fees Privilige: none

Ownership : owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: changing swap threshold - enabling trades

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.

auditace.tech

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

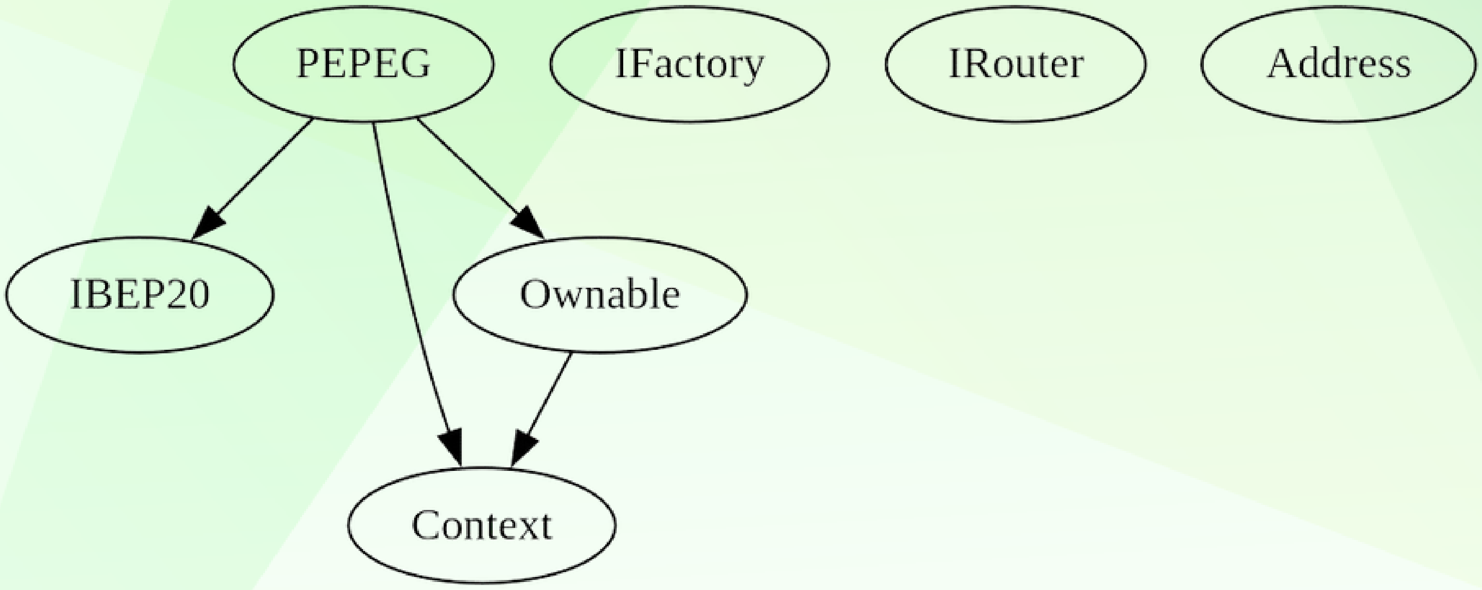
CLASSIFICATION OF RISK

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	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change fees (5% for each type of tax)**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- Owner must enable trades manually

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❗️
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗️	🛑	NO❗️
└	owner	Public ❗️		NO❗️
└	renounceOwnership	Public ❗️	🛑	onlyOwner
└	transferOwnership	Public ❗️	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗️	🛑	NO❗️
IRouter	Interface			
└	factory	External ❗️		NO❗️
└	WETH	External ❗️		NO❗️
└	addLiquidityETH	External ❗️	💵	NO❗️
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗️	🛑	NO❗️
Address	Library			
└	sendValue	Internal 🔒	🛑	
PEPEG	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗️	🛑	NO❗️
└	name	Public ❗️		NO❗️
└	symbol	Public ❗️		NO❗️
└	decimals	Public ❗️		NO❗️
└	totalSupply	Public ❗️		NO❗️
└	balanceOf	Public ❗️		NO❗️

CONTRACT ASSESMENT

```
| └ | allowance | Public ❗ |   |NO❗ |
| └ | approve | Public ❗ | 🛑  |NO❗ |
| └ | transferFrom | Public ❗ | 🛑  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | transfer | Public ❗ | 🛑  |NO❗ |
| └ | isExcludedFromReward | Public ❗ |   |NO❗ |
| └ | reflectionFromToken | Public ❗ |   |NO❗ |
| └ | EnableTrading | External ❗ | 🛑  | onlyOwner |
| └ | updatedeadline | External ❗ | 🛑  | onlyOwner |
| └ | tokenFromReflection | Public ❗ |   |NO❗ |
| └ | excludeFromReward | Public ❗ | 🛑  | onlyOwner |
| └ | includeInReward | External ❗ | 🛑  | onlyOwner |
| └ | excludeFromFee | Public ❗ | 🛑  | onlyOwner |
| └ | includeInFee | Public ❗ | 🛑  | onlyOwner |
| └ | isExcludedFromFee | Public ❗ |   |NO❗ |
| └ | _reflectRfi | Private 🔐 | 🛑  | |
| └ | _takeLiquidity | Private 🔐 | 🛑  | |
| └ | _takeMarketing | Private 🔐 | 🛑  | |
| └ | _takeOps | Private 🔐 | 🛑  | |
| └ | _takeDev | Private 🔐 | 🛑  | |
| └ | _getValues | Private 🔐 |   | |
| └ | _getTValues | Private 🔐 |   | |
| └ | _getRValues1 | Private 🔐 |   | |
| └ | _getRValues2 | Private 🔐 |   | |
| └ | _getRate | Private 🔐 |   | |
| └ | _getCurrentSupply | Private 🔐 |   | |
| └ | _approve | Private 🔐 | 🛑  | |
| └ | _transfer | Private 🔐 | 🛑  | |
| └ | _tokenTransfer | Private 🔐 | 🛑  | |
| └ | swapAndLiquify | Private 🔐 | 🛑  | lockTheSwap |
| └ | addLiquidity | Private 🔐 | 🛑  | |
| └ | swapTokensForBNB | Private 🔐 | 🛑  | |
| └ | bulkExcludeFee | External ❗ | 🛑  | onlyOwner |
| └ | updateMarketingWallet | External ❗ | 🛑  | onlyOwner |
| └ | updateDevWallet | External ❗ | 🛑  | onlyOwner |
| └ | updateOpsWallet | External ❗ | 🛑  | onlyOwner |
| └ | updateSwapTokensAtAmount | External ❗ | 🛑  | onlyOwner |
| └ | updateSwapEnabled | External ❗ | 🛑  | onlyOwner |
| └ | rescueBNB | External ❗ | 🛑  | onlyOwner |
| └ | rescueAnyBEP20Tokens | Public ❗ | 🛑  | onlyOwner |
```

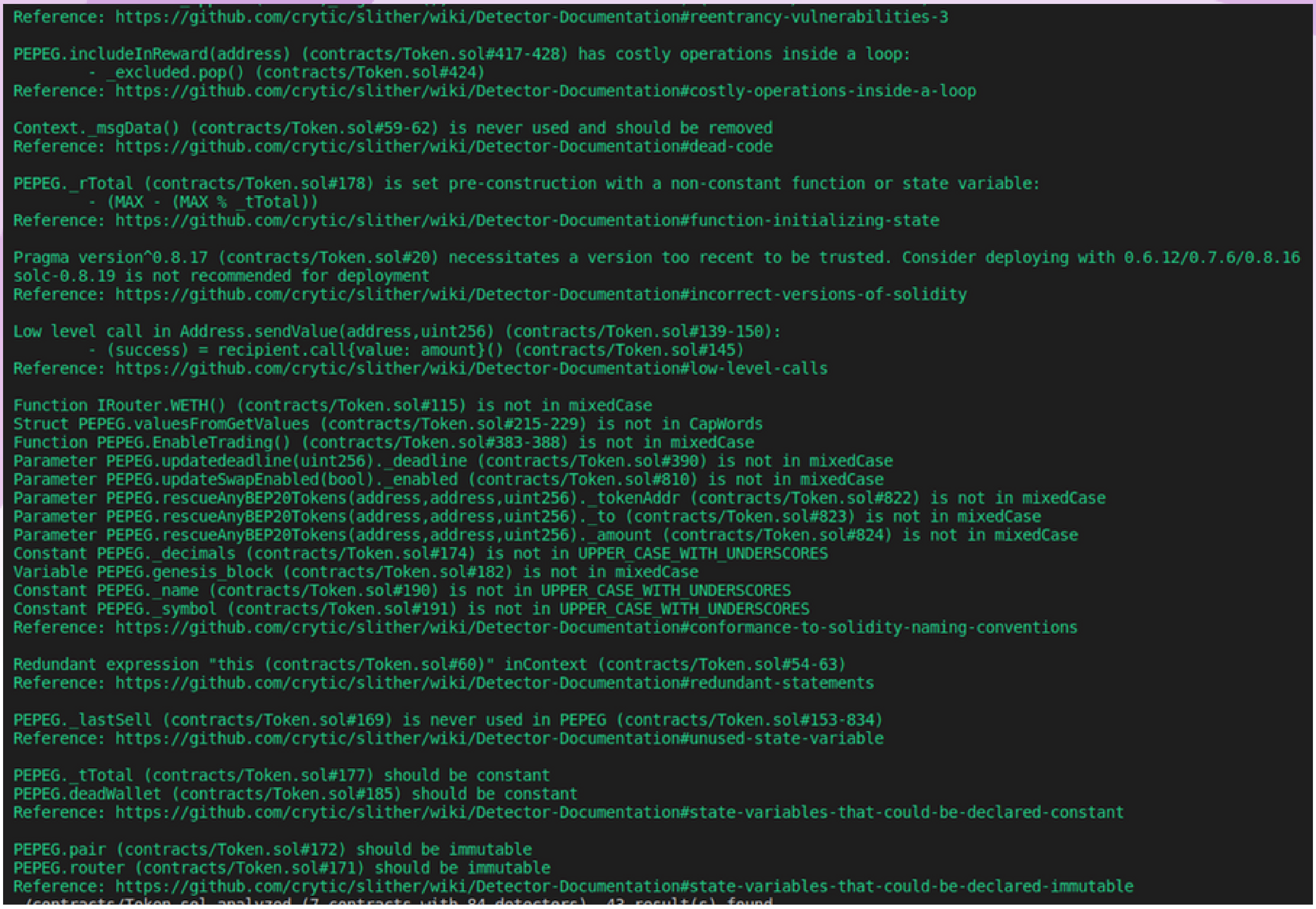
CONTRACT ASSESMENT

| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

PEPEG.includeInReward(address) (contracts/Token.sol#417-428) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#424)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#59-62) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

PEPEG._rTotal (contracts/Token.sol#178) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#20) 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

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#139-150):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#115) is not in mixedCase
Struct PEPEG.valuesFromGetValues (contracts/Token.sol#215-229) is not in CapWords
Function PEPEG.EnableTrading() (contracts/Token.sol#383-388) is not in mixedCase
Parameter PEPEG.updatedeadline(uint256)._deadline (contracts/Token.sol#390) is not in mixedCase
Parameter PEPEG.updateSwapEnabled(bool)._enabled (contracts/Token.sol#810) is not in mixedCase
Parameter PEPEG.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#822) is not in mixedCase
Parameter PEPEG.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#823) is not in mixedCase
Parameter PEPEG.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#824) is not in mixedCase
Constant PEPEG._decimals (contracts/Token.sol#174) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PEPEG.genesis_block (contracts/Token.sol#182) is not in mixedCase
Constant PEPEG._name (contracts/Token.sol#190) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPEG._symbol (contracts/Token.sol#191) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#60)" inContext (contracts/Token.sol#54-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PEPEG._lastSell (contracts/Token.sol#169) is never used in PEPEG (contracts/Token.sol#153-834)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

PEPEG._tTotal (contracts/Token.sol#177) should be constant
PEPEG.deadWallet (contracts/Token.sol#185) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PEPEG.pair (contracts/Token.sol#172) should be immutable
PEPEG.router (contracts/Token.sol#171) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
 (contracts/Token.sol analyzed (7 contracts with 84 detectors), 43 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

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/0xbff3d2ee08bc34c7c4ee87092c78971f101c9446cfe1b46791ee64638e238fdf

2- Buying when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0xa35404088c75c6fd0cc763a0c1f58a9625543a8aab0f596c1dc6d4c8e0362b56

3- Selling when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0x463f66297b155b371fcf24d89a5341f2812e351be7e62c852f4c583d54b06f84

4- Transferring when excluded (0% tax) **(passed):**
https://testnet.bscscan.com/tx/0xa8bb9ef5114b0d8e7edb423902ead674f9038dbc16c23589983b4f9415b41e59

5- Buying when not excluded (5% tax) **(passed):**
https://testnet.bscscan.com/tx/0x8b6262b3190e0c2426313d5b53dca24a37e21c6c92e3c7201e8da7fe19eccb36

6- Selling when not excluded (5% tax) **(passed):**
https://testnet.bscscan.com/tx/0x634f6cbeb1c5e3b29869628385616cf622670cf1223aa69720a49ea41f89aa70

FUNCTIONAL TESTING

7- Transferring when not excluded (5% tax) (passed):
https://testnet.bscscan.com/tx/0x61870d3f8806e5410d01c137212a22db3f75235f9d587defc6880e42a337f66c

8- Internal swap (marketing + liquidity) (passed):
https://testnet.bscscan.com/tx/0x634f6cbeb1c5e3b29869628385616cf622670cf1223aa69720a49ea41f89aa70

MANUAL TESTING

Centralization – Trades must be enabled

Severity: Medium

function: EnableTrading

Status: Resolved (Safu dev)

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    swapEnabled = true;
    genesis_block = block.number;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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