auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PEPE PINK

DATED : 6 July 23'

AUDIT SUMMARY

Project name – PEPE PINK

Date: 6 July, 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	1	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/0xEf7b6F7672c34da2Fa7F1B4649c457566Fc0BADC

Token Information

Token Name : PEPE PINK

Token Symbol: PEPINK

Decimals: 9

Token Supply: 420,690,000,000,000

Token Address:
0x4BE8DFb2De1b63051bf159e699dF5941F17f0388

Checksum:
eb3e8e8c1bc4f5dea1799b25ae684fed19e124cc

Owner:
0x8445661670dB1c2E31A41401c3cA0d5b9Aa6925A
(at time of writing the audit)

Deployer:
0x8445661670dB1c2E31A41401c3cA0d5b9Aa6925A

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 1%

Fees Privilege: no fees

Ownership: renounced

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of the tokens 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.

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

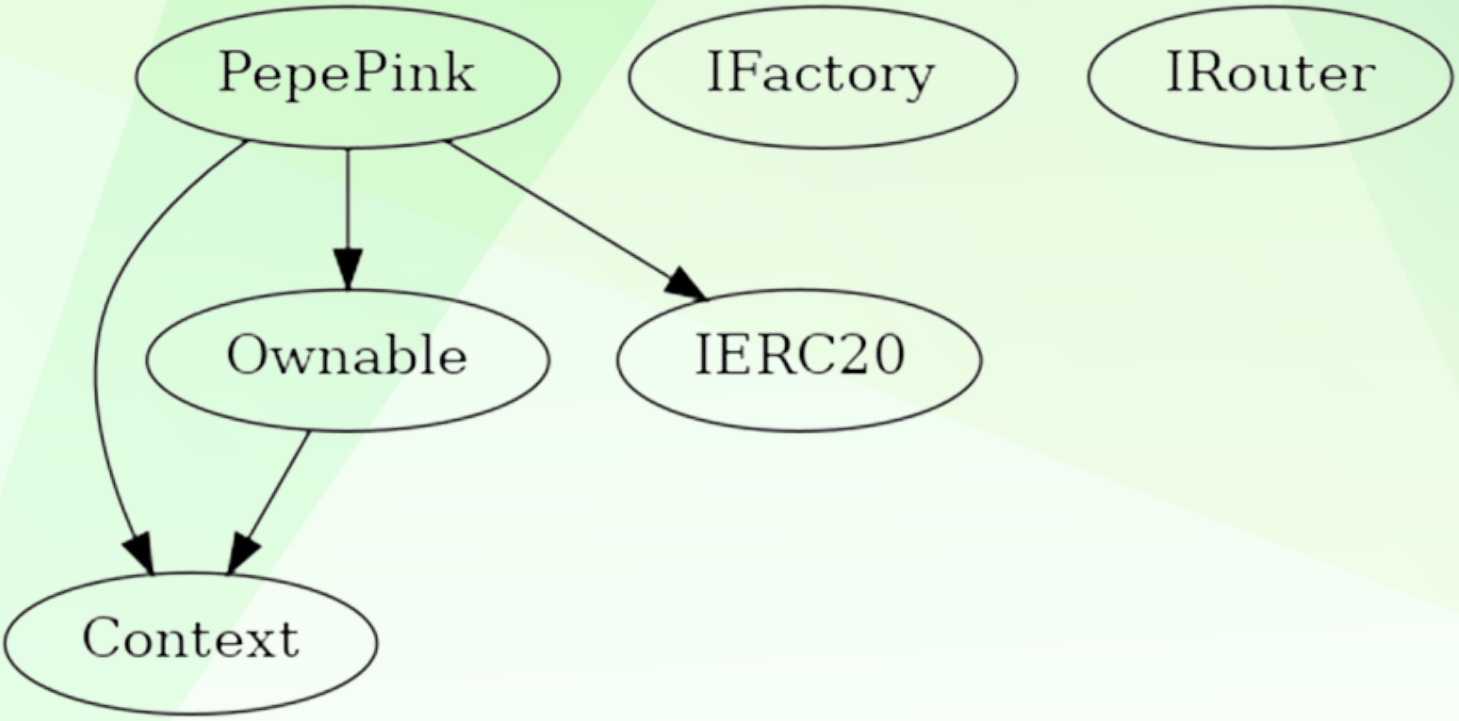
CLASSIFICATION OF RISK

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

INHERITANCE TREE



CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IERC20Standadard	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20Standadard		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	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 ❗
IUniswapPair	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗

CONTRACT ASSESMENT

└	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 ❗	
└	swap	External ❗	🛑	NO ❗	
└	skim	External ❗	🛑	NO ❗	
└	sync	External ❗	🛑	NO ❗	
└	initialize	External ❗	🛑	NO ❗	
IUniswapRouter01	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 ❗	

CONTRACT ASSESMENT

└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IUniswapRouter02	Interface	IUniswapRouter01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
ERC20	Implementation	Context, IERC20Standadard, 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 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
AUTH	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	fee	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
SafeMathInt	Library			
└	mul	Internal 🔒		

CONTRACT ASSESMENT

└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
DividendInterface	Interface				
└	withdrawableDividendOf	External ❗		NO ❗	
└	withdrawnDividendOf	External ❗		NO ❗	
└	accumulativeDividendOf	External ❗	🛑	NO ❗	
SafeMathUint	Library				
└	toInt256Safe	Internal 🔒		
PePa2	Implementation	ERC20, Ownable, AUTH			
└	<Constructor>	Public ❗	💵	ERC20
└	removeLimits	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	amountExcludedFromTax	Internal 🔒	🛑	
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	swapToFeeBack	Private 🔐	🛑	
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

POINTS TO NOTE

- **Owner is not able to change current fees (1% buy / 1% sell / 1% transfer)**
- Owner is not able to blacklist an address
- Owner is not able to disable buy/sell/transfers
- Owner is not able to set max wallet limit and minimum wallet limits
- Owner is not able to mint new tokens
- **Owner must enable trades manually**

STATIC ANALYSIS

```
Reentrancy in PepePink.transferFrom(address,address,uint256) (contracts/Token.sol#237-245):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#238)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#491-499)
                - (success,None) = marketingWallet.call{gas: 35000,value: address(this).balance}() (contracts/Token.sol#478)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#238)
                - (success,None) = marketingWallet.call{gas: 35000,value: address(this).balance}() (contracts/Token.sol#478)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#411)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/Token.sol#242)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

PepePink.includeInReward(address) (contracts/Token.sol#303-315) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#310)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#27-30) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#17) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in PepePink.swapAndLiquify() (contracts/Token.sol#469-480):
        - (success,None) = marketingWallet.call{gas: 35000,value: address(this).balance}() (contracts/Token.sol#478)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#75) is not in mixedCase
Struct PepePink.valuesFromGetValues (contracts/Token.sol#167-175) is not in CapWords
Event PepePink_tradingEnabled() (contracts/Token.sol#149) is not in CapWords
Event PepePinkexcludeFromRewardAccount(address) (contracts/Token.sol#286) is not in CapWords
Event PepePinkincludeInRewardAccount(address) (contracts/Token.sol#301) is not in CapWords
Event PepePinkexcludeFromFeeWallet(address) (contracts/Token.sol#317) is not in CapWords
Event PepePinkincludeInFeeWallet(address) (contracts/Token.sol#326) is not in CapWords
Event PepePinkupdateMarketingWallet(address) (contracts/Token.sol#502) is not in CapWords
Event PepePink_updateSwapTokensAtAmount(uint256) (contracts/Token.sol#511) is not in CapWords
Constant PepePink._decimals (contracts/Token.sol#134) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepePink._tTotal (contracts/Token.sol#137) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepePink._name (contracts/Token.sol#146) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepePink._symbol (contracts/Token.sol#147) 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#28)" inContext (contracts/Token.sol#20-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PepePink.deployerWallet (contracts/Token.sol#144) should be immutable
PepePink.pair (contracts/Token.sol#132) should be immutable
PepePink.router (contracts/Token.sol#131) 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

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x73712c312fd3286024d6a6e2865fee80b0b1d31d1957b4cdc28e4a98363781c4

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

https://testnet.bscscan.com/tx/0x9f2da40ecdab4b64e3805e1be36a4338568d5150102b15a5372e96c7138a8234

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

https://testnet.bscscan.com/tx/0x8098fd83fe1b33ebaaf5f71bbb76f7324fb840c09f87a7c032b356bae34166db

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

https://testnet.bscscan.com/tx/0x58634768f7efecffc868c13287318782156b300d8382d51a990cde2679701662

5- Buying(1% tax) (passed):

https://testnet.bscscan.com/tx/0x70efc0f5cde5d8fb50c61ef66cae16d76c1e62b087b575b1cfbf205a837bf92a

6- Selling (1% tax) (passed):

https://testnet.bscscan.com/tx/0x0e8d98b820b968b2cbcef0905018466573491ac0e26231f442805e74aff942a6

FUNCTIONAL TESTING

4- Transferring (1% tax) (passed):

https://testnet.bscscan.com/tx/0xb1f318c8d34ada1d04c59ddd86a0936b23950eb15cbdc137bb8ef81750ac1033

4- Internal swap (passed):

https://testnet.bscscan.com/tx/0x0e8d98b820b968b2cbcef0905018466573491ac0e26231f442805e74aff942a6

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

function: enableTrading

Status: Resolved | Trades are Enabled

Overview:

Owner of the contract must enable trades manually for investors, otherwise no one would be able to buy/sell/transfer their tokens.

```
function enableTrading() external onlyOwner {
    require(!isTradingEnabled, "Trading already enabled");
    isTradingEnabled = true;
    emit _tradingEnabled();
}
```

Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a certified pinsksale safu developer to guearanee enabling of trades.

MANUAL TESTING

Access control – Unauthorized call to owner functions

Severity: Medium

function: clearStuckToken - manualSend

Status: Not Resolved

Overview:

calls to manualSend and clearStuckToken functions are not authorized meaning that a malicious actor can constantly call this function to send ETH or accumulated tokens back to deployerWallet which can disable internal swaps.

```
function manualSend() external {
    payable(deployerWallet).transfer(address(this).balance);
}

function clearStuckToken(address tokenAddress, uint256 tokens) external returns (bool success)
{
    if (tokens == 0) {
        tokens = IERC20(tokenAddress).balanceOf(address(this));
    }
    emit ClearToken(tokenAddress, tokens);
    return IERC20(tokenAddress).transfer(deployerWallet, tokens);
}
```

Suggestion

Ensure that both of this functions are onlyOwner

```
function manualSend() external onlyOwner{
    payable(deployerWallet).transfer(address(this).balance);
}

function clearStuckToken(address tokenAddress, uint256 tokens) external onlyOwner returns
(bool success) {
    if (tokens == 0) {
        tokens = IERC20(tokenAddress).balanceOf(address(this));
    }
    emit ClearToken(tokenAddress, tokens);
    return IERC20(tokenAddress).transfer(deployerWallet, tokens);
}
```

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