



# Smart Contract Audit

FOR  
**PEPE PINK**

DATED : 6 July 23'



# MANUAL TESTING

## Centralization – Enabling Trades

**Severity:** High

**function:** enableTrading

**Status:** Not Resolved

### 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

It's suggested to either enable trades prior to presale, or transfer ownership of the contract to a certified pinsksale safu developer to guarantee enabling of trades.



# 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 With High Risk**

## Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	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 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 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

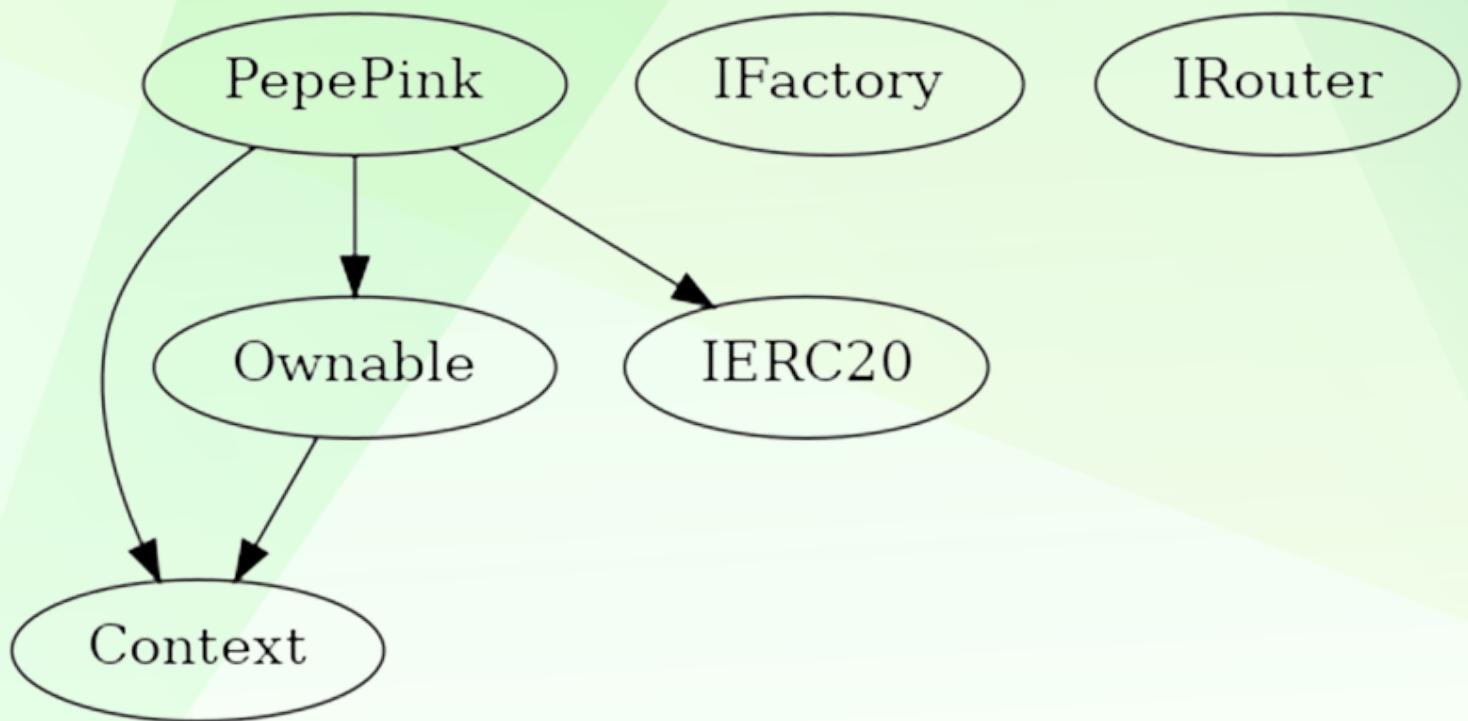
# CLASSIFICATION OF RISK

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

# INHERITANCE TREE





# CONTRACT ASSESSMENT

Contract	Type	Bases			
	L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
		**Context**	Implementation		
	L	_msgSender	Internal	!	
	L	_msgData	Internal	!	
		**SafeMath**	Library		
	L	add	Internal	!	
	L	sub	Internal	!	
	L	sub	Internal	!	
	L	mul	Internal	!	
	L	div	Internal	!	
	L	div	Internal	!	
	L	mod	Internal	!	
	L	mod	Internal	!	
		**IERC20Standadard**	Interface		
	L	totalSupply	External	!	!
	L	balanceOf	External	!	!
	L	transfer	External	!	!
	L	allowance	External	!	!
	L	approve	External	!	!
	L	transferFrom	External	!	!
		**IERC20Metadata**	Interface	IERC20Standadard	
	L	name	External	!	!
	L	symbol	External	!	!
	L	decimals	External	!	!
		**IUniswapV2Factory**	Interface		
	L	feeTo	External	!	!
	L	feeToSetter	External	!	!
	L	getPair	External	!	!
	L	allPairs	External	!	!
	L	allPairsLength	External	!	!
	L	createPair	External	!	!
	L	setFeeTo	External	!	!
	L	setFeeToSetter	External	!	!
		**IUniswapPair**	Interface		
	L	name	External	!	!
	L	symbol	External	!	!



# CONTRACT ASSESSMENT

L   decimals   External !     NO !
L   totalSupply   External !     NO !
L   balanceOf   External !     NO !
L   allowance   External !     NO !
L   approve   External !     ●   NO !
L   transfer   External !     ●   NO !
L   transferFrom   External !     ●   NO !
L   DOMAIN_SEPARATOR   External !     NO !
L   PERMIT_TYPEHASH   External !     NO !
L   nonces   External !     NO !
L   permit   External !     ●   NO !
L   MINIMUM_LIQUIDITY   External !     NO !
L   factory   External !     NO !
L   token0   External !     NO !
L   token1   External !     NO !
L   getReserves   External !     NO !
L   price0CumulativeLast   External !     NO !
L   price1CumulativeLast   External !     NO !
L   kLast   External !     NO !
L   mint   External !     ●   NO !
L   burn   External !     ●   NO !
L   swap   External !     ●   NO !
L   skim   External !     ●   NO !
L   sync   External !     ●   NO !
L   initialize   External !     ●   NO !
**IUniswapRouter01**   Interface
L   factory   External !     NO !
L   WETH   External !     NO !
L   addLiquidity   External !     ●   NO !
L   addLiquidityETH   External !     \$   NO !
L   removeLiquidity   External !     ●   NO !
L   removeLiquidityETH   External !     ●   NO !
L   removeLiquidityWithPermit   External !     ●   NO !
L   removeLiquidityETHWithPermit   External !     ●   NO !
L   swapExactTokensForTokens   External !     ●   NO !
L   swapTokensForExactTokens   External !     ●   NO !
L   swapExactETHForTokens   External !     \$   NO !
L   swapTokensForExactETH   External !     ●   NO !
L   swapExactTokensForETH   External !     ●   NO !
L   swapETHForExactTokens   External !     \$   NO !
L   quote   External !     NO !

# CONTRACT ASSESSMENT

```
| L | getAmountOut | External ! | NO ! | | |
| L | getAmountIn | External ! | NO ! |
| L | getAmountsOut | External ! | NO ! |
| L | getAmountsIn | External ! | NO ! |
||||
| **IUniswapRouter02** | Interface | IUniswapRouter01 ||
| L | removeLiquidityETHSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ! | ● | NO ! |
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ! | $ | NO ! |
| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External ! | ● | NO ! |
||||
| **ERC20** | Implementation | Context, IERC20Standadard, IERC20Metadata ||
| L | <Constructor> | Public ! | ● | NO ! |
| L | name | Public ! | | NO ! |
| L | symbol | Public ! | | NO ! |
| L | decimals | Public ! | | NO ! |
| L | totalSupply | Public ! | | NO ! |
| L | balanceOf | Public ! | | NO ! |
| L | transfer | Public ! | ● | NO ! |
| L | allowance | Public ! | | NO ! |
| L | approve | Public ! | ● | NO ! |
| L | transferFrom | Public ! | ● | NO ! |
| L | increaseAllowance | Public ! | ● | NO ! |
| L | decreaseAllowance | Public ! | ● | NO ! |
| L | _transfer | Internal 🔒 | ● |||
| L | _mint | Internal 🔒 | ● |||
| L | _burn | Internal 🔒 | ● |||
| L | _approve | Internal 🔒 | ● |||
| L | _beforeTokenTransfer | Internal 🔒 | ● |||
||||
| **AUTH** | Implementation | ||
| L | <Constructor> | Public ! | ● | NO ! |
| L | fee | Internal 🔒 | |||
||||
| **Ownable** | Implementation | Context ||
| L | <Constructor> | Public ! | ● | NO ! |
| L | owner | Public ! | | NO ! |
| L | renounceOwnership | Public ! | | ● | onlyOwner |
| L | transferOwnership | Public ! | | ● | onlyOwner |
||||
| **SafeMathInt** | Library | ||
| L | mul | Internal 🔒 | |||
```



# CONTRACT ASSESSMENT

```
| L | div | Internal 🔒 | ||  
| L | sub | Internal 🔒 | ||  
| L | add | Internal 🔒 | ||  
| L | abs | Internal 🔒 | ||  
| L | toUInt256Safe | Internal 🔒 | ||  
|||||  
| **DividendInterface** | Interface | |||  
| L | withdrawableDividendOf | External ! | NO ! |  
| L | withdrawnDividendOf | External ! | NO ! |  
| L | accumulativeDividendOf | External ! | ● NO ! |  
|||||  
| **SafeMathUint** | Library | |||  
| L | toInt256Safe | Internal 🔒 | ||  
|||||  
| **PePa2** | Implementation | ERC20, Ownable, AUTH |||  
| L | <Constructor> | Public ! | $ | ERC20 |  
| L | removeLimits | External ! | ● | onlyOwner |  
| L | _transfer | Internal 🔒 | ● |  
| L | amountExcludedFromTax | Internal 🔒 | ● |  
| L | excludeFromFees | Public ! | ● | onlyOwner |  
| L | swapToFeeBack | Private 🔒 | ● |  
| L | _setAutomatedMarketMakerPair | Private 🔒 | ● |  
| L | swapTokensForEth | Private 🔒 | ● |  
| L | <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._total (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:** Not Resolved

### 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 guearantee 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 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>**

---