auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PepeFinance

DATED : 15 June 23'

MANUAL TESTING

Logical – 0% marketing fee can disable trades

Severity: Critical

function: _transfer

Status: Not Resolved

Overview:

Setting buyMarketingFee or sellMarketingFee to 0% can disable buy or sells. This is because this fees are subtracted by 1 at _transfer function.

```
if (takeFee) {
    // on buy
    if (isBuy && buyTotalFees > 0) {
        feeMarketing = amount.mul(buyMarketingFee.sub(1)).div(100);
        feeLiquidity = amount.mul(buyLiquidityFee.add(1)).div(100);
    }
    // on sell
    else if (isSell && sellTotalFees > 0) {
        feeMarketing = amount.mul(sellMarketingFee.sub(1)).div(100);
        feeLiquidity = amount.mul(sellLiquidityFee.add(1)).div(100);
    }
```

Suggestion

Before subtracting by 1, ensure that fees are greater than 0

```
if (takeFee) {
    // on buy
    if (isBuy && buyTotalFees > 0 && buyMarketingFee > 0) {
        feeMarketing = amount.mul(buyMarketingFee.sub(1)).div(100);
        feeLiquidity = amount.mul(buyLiquidityFee.add(1)).div(100);
    }
    // on sell
    else if (isSell && sellTotalFees > 0 && sellMarketingFee > 0) {
        feeMarketing = amount.mul(sellMarketingFee.sub(1)).div(100);
        feeLiquidity = amount.mul(sellLiquidityFee.add(1)).div(100);
    }
```

AUDIT SUMMARY

Project name – PepeFinance

Date: 15 June, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Failed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	1	0	1	0	1
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/0xca0883caff61E1C6f4751e3144567328D483782A

Token Information

Token Name : PepeFinance

Token Symbol: PepeF

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x0b310c1fAC9991BBA5aA30a7DFa72D66AAddB96b

Checksum:
4815b1bf56dcc984ff23f00769acfeb984eff34b

Owner:
0x10d9d74f0ACC91FF0B3e44c0Ff2fF079ec2E33f3

Deployer:
0x10d9d74f0ACC91FF0B3e44c0Ff2fF079ec2E33f3

TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of tokens updating 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
- Gasless Send
- Private modifier
- Using block.timestamp
- Multiple Sends
- Re-entrancy
- Using Suicide
- Tautology or contradiction
- Gas Limitand 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

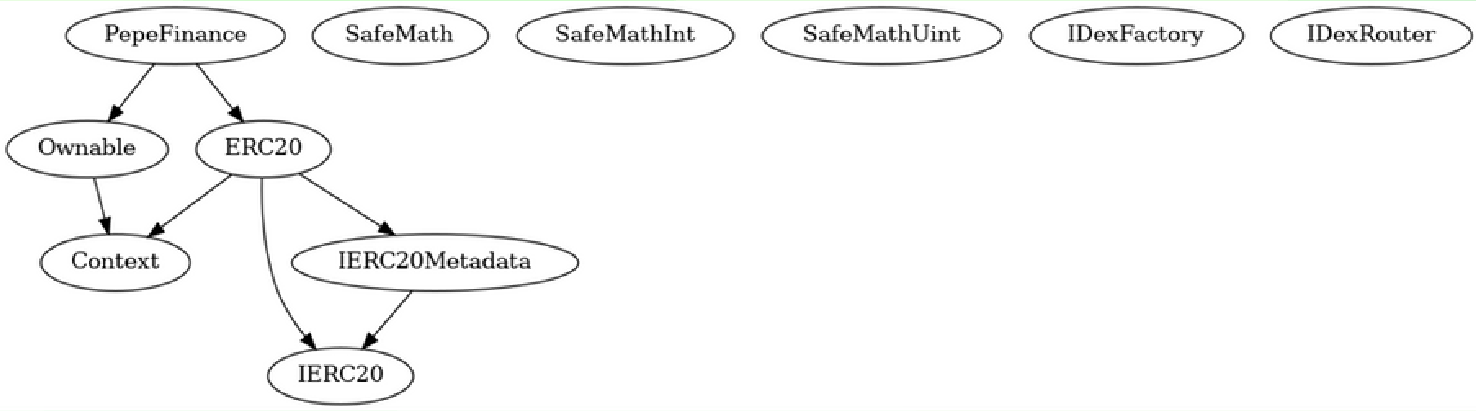
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to change buy/sell fees 0-10%
- 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

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗️		NO ❗️
└	balanceOf	External ❗️		NO ❗️
└	transfer	External ❗️	🛑	NO ❗️
└	allowance	External ❗️		NO ❗️
└	approve	External ❗️	🛑	NO ❗️
└	transferFrom	External ❗️	🛑	NO ❗️
IERC20Metadata	Interface	IERC20		
└	name	External ❗️		NO ❗️
└	symbol	External ❗️		NO ❗️
└	decimals	External ❗️		NO ❗️
ERC20	Implementation	Context, IERC20, 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 🔒	🛑	
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		

CONTRACT ASSESMENT

└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗️	🛑	NO ❗️
└	owner	Public ❗️		NO ❗️
└	renounceOwnership	Public ❗️	🛑	onlyOwner
└	transferOwnership	Public ❗️	🛑	onlyOwner
SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		
IDexFactory	Interface			
└	getPair	External ❗️		NO ❗️
└	createPair	External ❗️	🛑	NO ❗️
IDexRouter	Interface			
└	factory	External ❗️		NO ❗️
└	WETH	External ❗️		NO ❗️
└	addLiquidityETH	External ❗️	💵	NO ❗️
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗️	🛑	NO ❗️
PepeFinance	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗️	🛑	ERC20
└	<Receive Ether>	External ❗️	💵	NO ❗️
└	updateSwapTokensAtAmount	External ❗️	🛑	onlyOwner
└	updateBuyFees	External ❗️	🛑	onlyOwner
└	updateSellFees	External ❗️	🛑	onlyOwner
└	excludeFromMaxTransaction	Public ❗️	🛑	onlyOwner
└	updateSwapBackEnabled	External ❗️	🛑	onlyOwner
└	excludeFromFees	Public ❗️	🛑	onlyOwner

CONTRACT ASSESMENT

└	recoverETH	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	isExcludedFromFees	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	swapBack	Private 🔐	🛑	
└	burn	External ❗	🛑	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in PepeFinance.swapBack() (contracts/Token.sol#920-949):
	External calls:
	- swapTokensForEth(amountToSwapForETH) (contracts/Token.sol#932)
		- dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#896-902)
	- addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#944)
		- dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/Token.sol#910-917)
	External calls sending eth:
	- addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#944)
		- dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/Token.sol#910-917)
	Event emitted after the call(s):
	- Approval(owner,spender,amount) (contracts/Token.sol#365)
		- addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#944)
	- SwapAndLiquify(amountToSwapForETH,ethForLiquidity,tokensForLiquidity) (contracts/Token.sol#945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Token.sol#20-23) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#506-508) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#522-525) is never used and should be removed
SafeMathInt.abs(int256) (contracts/Token.sol#628-631) is never used and should be removed
SafeMathInt.add(int256,int256) (contracts/Token.sol#619-623) is never used and should be removed
SafeMathInt.div(int256,int256) (contracts/Token.sol#599-605) is never used and should be removed
SafeMathInt.mul(int256,int256) (contracts/Token.sol#587-594) is never used and should be removed
SafeMathInt.sub(int256,int256) (contracts/Token.sol#610-614) is never used and should be removed
SafeMathInt.toUint256Safe(int256) (contracts/Token.sol#633-636) is never used and should be removed
SafeMathUint.toInt256Safe(uint256) (contracts/Token.sol#640-644) 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#6) 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 PepeFinance.recoverETH() (contracts/Token.sol#799-801):
	- address(msg.sender).call{value: address(this).balance}() (contracts/Token.sol#800)
Low level call in PepeFinance.swapBack() (contracts/Token.sol#920-949):
	- (success) = address(_taxWallet.marketing).call{value: address(this).balance}() (contracts/Token.sol#948)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IDexRouter.WETH() (contracts/Token.sol#654) is not in mixedCase
Parameter PepeFinance.updateBuyFees(uint256,uint256)._marketingFee (contracts/Token.sol#771) is not in mixedCase
Parameter PepeFinance.updateBuyFees(uint256,uint256)._liquidityFee (contracts/Token.sol#771) is not in mixedCase
Parameter PepeFinance.updateSellFees(uint256,uint256)._marketingFee (contracts/Token.sol#778) is not in mixedCase
Parameter PepeFinance.updateSellFees(uint256,uint256)._liquidityFee (contracts/Token.sol#778) is not in mixedCase
Constant PepeFinance.deadAddress (contracts/Token.sol#679) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PepeFinance._taxWallet (contracts/Token.sol#691) is not in mixedCase
Variable PepeFinance._isExcludedMaxTransactionAmount (contracts/Token.sol#710) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#21)" inContext (contracts/Token.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PepeFinance.updateSwapTokensAtAmount(uint256) (contracts/Token.sol#765-769) uses literals with too many digits:
	- require(bool,string)(newAmount >= totalSupply() * 1 / 100000,Swap amount cannot be lower than 0.001% total supply.) (contracts/Token.sol#766)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/Token.sol#582) is never used in SafeMathInt (contracts/Token.sol#580-637)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

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 (trades disabled) (passed):
https://testnet.bscscan.com/tx/0x28de832ecbc300f0ac8899aea74b3e47e68a5e0e0a49ea820ccb70f185e6698b

2- Buying when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x0ae01fcf60f9b7a3c36e3317dd819fe194db98fe5ebe6ccc9de19b94854718b3

3- Selling when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xeedf88796dd5b6d4140f830a2ed59d30975e9ff5a1d82dc1acc007387c524a23

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5eef44896a4a651067fe67b9c3fe7a5524e4dd666d0bcce2073d3b34afaaa5b7

5- Buying when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0xc20dca395bdc66fbe4d90d167e778cc096c7da9dd5617267f0fb917dd9202c2d

FUNCTIONAL TESTING

6- Selling when not excluded from fees (0-10% tax) **(passed):**

https://testnet.bscscan.com/tx/0xa0349372c2b4836054997d970e0d82af232a994b0f347f174fa4fecdbceb6d41

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

https://testnet.bscscan.com/tx/0xf2ff93a73927abe1da2d92cc4a7e08359f282ef85f410a771056fd14b8a27a6b

8- Internal swap & rewards distribution **(passed):**

- BNB sent to marketing wallet

- Auto-liqudity

https://testnet.bscscan.com/tx/0xa0349372c2b4836054997d970e0d82af232a994b0f347f174fa4fecdbceb6d41

MANUAL TESTING

Logical – 0% marketing fee can disable trades

Severity: Critical

function: _transfer

Status: Not Resolved

Overview:

Setting buyMarketingFee or sellMarketingFee to 0% can disable buy or sells. This is because this fees are subtracted by 1 at _transfer function.

```
if (takeFee) {
   // on buy
   if (isBuy && buyTotalFees > 0) {
      feeMarketing = amount.mul(buyMarketingFee.sub(1)).div(100);
      feeLiquidity = amount.mul(buyLiquidityFee.add(1)).div(100);
   }
   // on sell
   else if (isSell && sellTotalFees > 0) {
      feeMarketing = amount.mul(sellMarketingFee.sub(1)).div(100);
      feeLiquidity = amount.mul(sellLiquidityFee.add(1)).div(100);
   }
```

Suggestion

Before subtracting by 1, ensure that fees are greater than 0

```
if (takeFee) {
   // on buy
   if (isBuy && buyTotalFees > 0 && buyMarketingFee > 0) {
      feeMarketing = amount.mul(buyMarketingFee.sub(1)).div(100);
      feeLiquidity = amount.mul(buyLiquidityFee.add(1)).div(100);
   }
   // on sell
   else if (isSell && sellTotalFees > 0 && sellMarketingFee > 0) {
      feeMarketing = amount.mul(sellMarketingFee.sub(1)).div(100);
      feeLiquidity = amount.mul(sellLiquidityFee.add(1)).div(100);
  }
```

MANUAL TESTING

Centralization – Owner receiving LP tokens

Severity: Medium

function: addLiquidity

Status: Not Resolved

Overview:

Owner is receiving LP tokens generated from auto-liquidity. This accumulated tokens can be used to remove a portion of liquidity pool.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(dexRouter), tokenAmount);

    // add the liquidity
    dexRouter.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

Suggestion

its suggeted either Lock or burn new LP tokens.

MANUAL TESTING

Informational – Locked ERC20 tokens

function: ---

Status: Not Resolved

Overview:

Currently there are no functions to withdraw stuck ERC20 tokens from the contract.

Suggestion

its suggeted to implement a function for withdrawing 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