auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PEPE GROK

DATED : 19 Dec 23'

MANUAL TESTING

Centralization – Buy and Sell Fees.

Severity: High

function: setBuyFee and setSellFee

Status: Open

Overview:

The owner can set the buy and sell fees to more than 100%, which is not recommended.

```
function setBuyFee(uint256 bf) external onlyOwner{
    buyFee = bf;
 }
 function setSellFee(uint256 sf) external onlyOwner{
    sellFee = sf;
 }
```

Suggestion

It is recommended that no fees in the contract should be more than 25% of the contract.

MANUAL TESTING

Centralization – The owner can Blacklist Wallet.

Severity: High

function: blacklistAddress

Status: Open

Overview:

The owner can blacklist multiple wallets.

```
function blacklistAddress(address account, bool value) public
onlyOwner {
    _isBlacklisted[account] = value;
  }
```

AUDIT SUMMARY

Project name – PEPE GROK

Date: 19 Dec, 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	0	2	0	2	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/address/0x418c56591bf6e834d4fedd0dde9356f4863f1699#code

Token Information

Token Address:
0xeebCAE2F8aBFEA67f42E7b2B18b8B7b56628EB21

Name: PEPE GROK

Symbol: PEPE GROK

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0x2DB68BEd43a06F7C164A8531d63E2163D7Ad863C

Deployer:
0x2DB68BEd43a06F7C164A8531d63E2163D7Ad863C

Total Supply: 420,690,000,000,000

Checksum: ade3cef7c2c788bc03532d7342fc9fak

Testnet:
https://testnet.bscscan.com/address/0x418c56591bf6e834d4fedd0dde9356f4863f1699#code

TOKEN OVERVIEW

Buy Fee: 0-100%

Sell Fee: 0-100%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

Blacklist: Yes

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
- 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

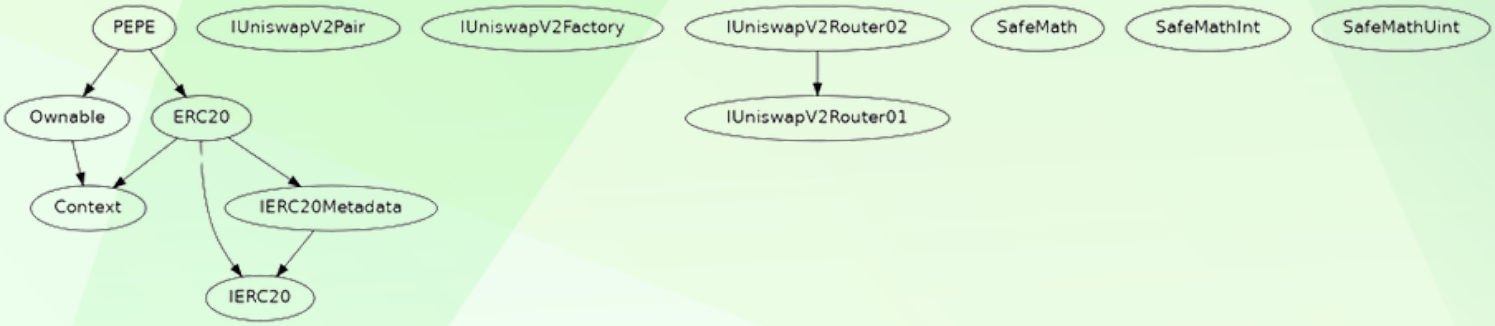
CLASSIFICATION OF RISK

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	2
Medium-Risk	0
Low-Risk	2
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

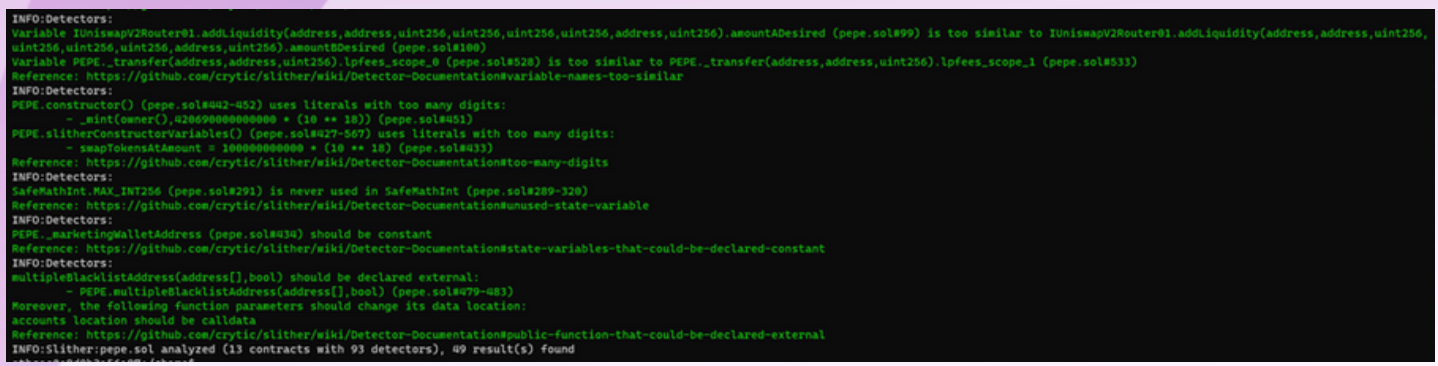
- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can exclude/include wallets from fees.
- The owner can set buy and sell fees of more than 100%.
- The owner can set swap Amounts.
- The owner can blacklist multiple wallet addresses.

STATIC ANALYSIS

```
INFO:Detectors:
Contract locking ether found:
        Contract PEPE (pepe.sol#427-567) has payable functions:
         - PEPE.receive() (pepe.sol#453)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
Reentrancy in PEPE._transfer(address,address,uint256) (pepe.sol#487-539):
        External calls:
        - _swapAndLiquid() (pepe.sol#516)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,_marketingWalletAddress,block.timestamp) (pepe.sol#547-553)
        State variables written after the call(s):
        - super._transfer(from,address(this),lpfees_scope_0) (pepe.sol#530)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (pepe.sol#392)
                - _balances[recipient] = _balances[recipient].add(amount) (pepe.sol#393)
        ERC20._balances (pepe.sol#332) can be used in cross function reentrancies:
        - ERC20._mint(address,uint256) (pepe.sol#396-402)
        - ERC20._transfer(address,address,uint256) (pepe.sol#384-395)
        - ERC20.balanceOf(address) (pepe.sol#353-355)
        - super._transfer(from,address(this),lpfees_scope_1) (pepe.sol#535)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (pepe.sol#392)
                - _balances[recipient] = _balances[recipient].add(amount) (pepe.sol#393)
        ERC20._balances (pepe.sol#332) can be used in cross function reentrancies:
        - ERC20._mint(address,uint256) (pepe.sol#396-402)
        - ERC20._transfer(address,address,uint256) (pepe.sol#384-395)
        - ERC20.balanceOf(address) (pepe.sol#353-355)
        - super._transfer(from,to,amount) (pepe.sol#538)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (pepe.sol#392)
                - _balances[recipient] = _balances[recipient].add(amount) (pepe.sol#393)
        ERC20._balances (pepe.sol#332) can be used in cross function reentrancies:
        - ERC20._mint(address,uint256) (pepe.sol#396-402)
        - ERC20._transfer(address,address,uint256) (pepe.sol#384-395)
        - ERC20.balanceOf(address) (pepe.sol#353-355)
        - swapping = false (pepe.sol#517)
        PEPE.swapping (pepe.sol#431) can be used in cross function reentrancies:
        - PEPE._transfer(address,address,uint256) (pepe.sol#487-539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
PEPE.setSwapAmounts(uint256) (pepe.sol#458-460) should emit an event for:
        - swapTokensAtAmount = value (pepe.sol#459)
PEPE.settimes(uint256) (pepe.sol#461-463) should emit an event for:
        - times = t (pepe.sol#462)
PEPE.settimeSecord(uint256) (pepe.sol#464-466) should emit an event for:
        - timeSecord = tt (pepe.sol#465)
PEPE.settimeFee(uint256) (pepe.sol#467-469) should emit an event for:
        - timeFee = ttt (pepe.sol#468)
PEPE.setBuyFee(uint256) (pepe.sol#470-472) should emit an event for:
        - buyFee = bf (pepe.sol#471)
PEPE.setSellFee(uint256) (pepe.sol#473-475) should emit an event for:
        - sellFee = sf (pepe.sol#474)
```

```
INFO:Detectors:
Reentrancy in PEPE._transfer(address,address,uint256) (pepe.sol#487-539):
        External calls:
        - _swapAndLiquid() (pepe.sol#516)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,_marketingWalletAddress,block.timestamp) (pepe.sol#547-553)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (pepe.sol#394)
                - super._transfer(from,to,amount) (pepe.sol#538)
        - Transfer(sender,recipient,amount) (pepe.sol#394)
                - super._transfer(from,address(this),lpfees_scope_0) (pepe.sol#530)
        - Transfer(sender,recipient,amount) (pepe.sol#394)
                - super._transfer(from,address(this),lpfees_scope_1) (pepe.sol#535)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
PEPE._transfer(address,address,uint256) (pepe.sol#487-539) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= times,zero) (pepe.sol#501)
        - block.timestamp <= times.add(timeSecord) (pepe.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (pepe.sol#32-35) is never used and should be removed
ERC20._burn(address,uint256) (pepe.sol#403-409) is never used and should be removed
SafeMath.mod(uint256,uint256) (pepe.sol#280-282) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (pepe.sol#283-286) is never used and should be removed
SafeMathInt.abs(int256) (pepe.sol#312-315) is never used and should be removed
SafeMathInt.add(int256,int256) (pepe.sol#307-311) is never used and should be removed
SafeMathInt.div(int256,int256) (pepe.sol#298-301) is never used and should be removed
SafeMathInt.mul(int256,int256) (pepe.sol#292-297) is never used and should be removed
SafeMathInt.sub(int256,int256) (pepe.sol#302-306) is never used and should be removed
SafeMathInt.toUint256Safe(int256) (pepe.sol#316-319) is never used and should be removed
SafeMathUint.toInt256Safe(uint256) (pepe.sol#323-327) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.6.2 (pepe.sol#6) allows old versions
Pragma version^0.6.2 (pepe.sol#21) allows old versions
Pragma version^0.6.2 (pepe.sol#27) allows old versions
Pragma version^0.6.2 (pepe.sol#37) allows old versions
Pragma version^0.6.2 (pepe.sol#80) allows old versions
Pragma version^0.6.2 (pepe.sol#92) allows old versions
Pragma version^0.6.2 (pepe.sol#223) allows old versions
Pragma version^0.6.2 (pepe.sol#249) allows old versions
Pragma version^0.6.2 (pepe.sol#288) allows old versions
Pragma version^0.6.2 (pepe.sol#321) allows old versions
Pragma version^0.6.2 (pepe.sol#329) allows old versions
Pragma version^0.6.2 (pepe.sol#426) allows old versions
solc-0.6.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (pepe.sol#50) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (pepe.sol#51) is not in mixedCase
```

STATIC ANALYSIS



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- Approve (passed):

https://testnet.bscscan.com/tx/0x785a7f41a7c71f4bbed2689754f0399f3285a8efdeb5e13d7357e2e5aa9b0fc6

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0x6e9203084ce61d45dac559f2ac31dc17df28f06524ac30d55d69d5df2fe85aae

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0x43417efcd1611d854717b9fd98b354d55d2706f80dabce5f57ee8a8fe99754d1

4- Blacklist Address (passed):

https://testnet.bscscan.com/tx/0x4d311863901446c958d6f65dc13171adb89e616f748fb548614f12e0db18f618

MANUAL TESTING

Centralization – Buy and Sell Fees.

Severity: High

function: setBuyFee and setSellFee

Status: Open

Overview:

The owner can set the buy and sell fees to more than 100%, which is not recommended.

```
function setBuyFee(uint256 bf) external onlyOwner{
    buyFee = bf;
 }
 function setSellFee(uint256 sf) external onlyOwner{
    sellFee = sf;
 }
```

Suggestion

It is recommended that no fees in the contract should be more than 25% of the contract.

MANUAL TESTING

Centralization – The owner can Blacklist Wallet.

Severity: High

function: blacklistAddress

Status: Open

Overview:

The owner can blacklist multiple wallets.

```
function blacklistAddress(address account, bool value) public
onlyOwner {
    _isBlacklisted[account] = value;
  }
```

MANUAL TESTING

Centralization – Missing Events

Severity: Low

subject: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setSwapAmounts(uint256 value) external onlyOwner{
    swapTokensAtAmount = value;
  }
  function settimes(uint256 t) external onlyOwner{
    times = t;
  }
  function settimeSecord(uint256 tt) external onlyOwner{
    timeSecord = tt;
  }
   function settimeFee(uint256 ttt) external onlyOwner{
    timeFee = ttt;
  }
  function setBuyFee(uint256 bf) external onlyOwner{
    buyFee = bf;
  }
  function setSellFee(uint256 sf) external onlyOwner{
    sellFee = sf;
  }
```

MANUAL TESTING

Centralization – Old Compiler Version

Severity: Low

subject: Old Solidity version

Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.6.2;

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

MANUAL TESTING

Optimization

Severity: Optimization

subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them

```
function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating
bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
  }
```

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