auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

$PepeFace

DATED : 29 Nov 23'

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
    require(!swapTokenEnabled, "Cannot re-enable trading");
    swapTokenEnabled = true;
 }
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – $PepeFace

Date: 29 Nov, 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	0	1	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/0xa1bd04f5d60295ba60d0bce6c157bf1d7582ee05#code

Token Information

Token Address:
0x3E45C4f2FC4D2B505557449a66a024a42b5C34D2

Name: $PepeFace

Symbol: PPFACE

Decimals: 18

Network: Binance Smart Chain

Token Type: BEP-20

Owner:
0xd3bE8D24DFfe3Fe26cd47e37c7c9815c0E622fc2

Deployer:
0xd3bE8D24DFfe3Fe26cd47e37c7c9815c0E622fc2

Token Supply: 810000000000000000000000000000000

Checksum: afde641126e217b2b455d49e77fc4199

Testnet:
https://testnet.bscscan.com/address/0xa1bd04f5d60295ba60d0bce6c157bf1d7582ee05#code

TOKEN OVERVIEW

Buy Fee: 0-20%

Sell Fee: 0-20%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

Blacklist: No

Other Privileges:

-Whitelist to transfer without enabling trades

- 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

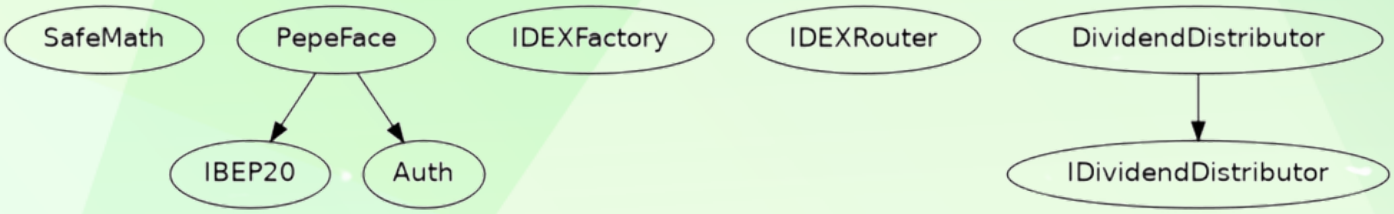
CLASSIFICATION OF RISK

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	0
Low-Risk	1
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- The owner can renounce ownership.
- The owner can transfer ownership.
- The owner can set a reward token.
- The owner can Enable trading.
- The owner can fee distribution.
- The owner can enable/disable swapping.

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311):
        External calls:
        - REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
        State variables written after the call(s):
        - shares[shareholder].totalRealised = shares[shareholder].totalRealised.add(amount) (PepeFace.sol#308)
        DividendDistributor.shares (PepeFace.sol#186) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311)
        - DividendDistributor.getUnpaidEarnings(address) (PepeFace.sol#317-326)
        - DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238)
        - DividendDistributor.shares (PepeFace.sol#186)
        - shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount) (PepeFace.sol#309)
        DividendDistributor.shares (PepeFace.sol#186) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311)
        - DividendDistributor.getUnpaidEarnings(address) (PepeFace.sol#317-326)
        - DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238)
        - DividendDistributor.shares (PepeFace.sol#186)
Reentrancy in DividendDistributor.process(uint256) (PepeFace.sol#269-293):
        External calls:
        - distributeDividend(investors[currentIndex]) (PepeFace.sol#285)
                - REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
        State variables written after the call(s):
        - currentIndex = 0 (PepeFace.sol#281)
        DividendDistributor.currentIndex (PepeFace.sol#198) can be used in cross function reentrancies:
        - DividendDistributor.process(uint256) (PepeFace.sol#269-293)
        - currentIndex ++ (PepeFace.sol#290)
        DividendDistributor.currentIndex (PepeFace.sol#198) can be used in cross function reentrancies:
        - DividendDistributor.process(uint256) (PepeFace.sol#269-293)
Reentrancy in DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238):
        External calls:
        - distributeDividend(shareholder) (PepeFace.sol#226)
                - REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
        State variables written after the call(s):
        - shares[shareholder].amount = amount (PepeFace.sol#236)
        DividendDistributor.shares (PepeFace.sol#186) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311)
        - DividendDistributor.getUnpaidEarnings(address) (PepeFace.sol#317-326)
        - DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238)
        - DividendDistributor.shares (PepeFace.sol#186)
        - shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount) (PepeFace.sol#237)
        DividendDistributor.shares (PepeFace.sol#186) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311)
        - DividendDistributor.getUnpaidEarnings(address) (PepeFace.sol#317-326)
        - DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238)
        - DividendDistributor.shares (PepeFace.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
PepeFace.swapBack() (PepeFace.sol#614-677) ignores return value by router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (PepeFace.sol#655-662)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
DividendDistributor.setDistributionCriteria(uint256,uint256) (PepeFace.sol#219-222) should emit an event for:
        - minPeriod = _minPeriod (PepeFace.sol#220)
        - minDistribution = _minDistribution (PepeFace.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311) has external calls inside a loop: REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in DividendDistributor.deposit() (PepeFace.sol#249-267):
        External calls:
        - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(0,path,address(this),block.timestamp) (PepeFace.sol#256-261)
        State variables written after the call(s):
        - dividendsPerShare = dividendsPerShare.add(dividendsPerShareAccuracyFactor.mul(amount).div(totalShares)) (PepeFace.sol#266)
        - totalDividends = totalDividends.add(amount) (PepeFace.sol#265)
Reentrancy in DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311):
        External calls:
        - REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
        State variables written after the call(s):
        - shareholderClaims[shareholder] = block.timestamp (PepeFace.sol#307)
Reentrancy in DividendDistributor.setShare(address,uint256) (PepeFace.sol#224-238):
        External calls:
        - distributeDividend(shareholder) (PepeFace.sol#226)
                - REWARD.transfer(shareholder,amount) (PepeFace.sol#306)
        State variables written after the call(s):
        - addShareholder(shareholder) (PepeFace.sol#230)
                - investors.push(shareholder) (PepeFace.sol#334)
        - removeShareholder(shareholder) (PepeFace.sol#232)
                - investors[shareholderIndexes[shareholder]] = investors[investors.length - 1] (PepeFace.sol#338)
                - investors.pop() (PepeFace.sol#340)
        - addShareholder(shareholder) (PepeFace.sol#230)
                - shareholderIndexes[shareholder] = investors.length (PepeFace.sol#333)
        - removeShareholder(shareholder) (PepeFace.sol#232)
                - shareholderIndexes[investors[investors.length - 1]] = shareholderIndexes[shareholder] (PepeFace.sol#339)
        - totalShares = totalShares.sub(shares[shareholder].amount).add(amount) (PepeFace.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

STATIC ANALYSIS

```
INFO:Detectors:
PepeFace.swapBack() (PepeFace.sol#614-677) tries to limit the gas of an external call that controls implicit decoding
        (success) = address(stakingReceiver).call{gas: 30000,value: amountBNBStaking}() (PepeFace.sol#642)
        (success_scope_0) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (PepeFace.sol#650)
        (success_scope_2) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNB_scope_1}() (PepeFace.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#return-bomb
INFO:Detectors:
DividendDistributor.shouldDistribute(address) (PepeFace.sol#295-298) uses timestamp for comparisons
        Dangerous comparisons:
        - shareholderClaims[shareholder] + minPeriod < block.timestamp && getUnpaidEarnings(shareholder) > minDistribution (PepeFace.sol#296-297)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
DividendDistributor.process(uint256) (PepeFace.sol#269-293) has costly operations inside a loop:
        - currentIndex = 0 (PepeFace.sol#281)
DividendDistributor.distributeDividend(address) (PepeFace.sol#300-311) has costly operations inside a loop:
        - totalDistributed = totalDistributed.add(amount) (PepeFace.sol#305)
DividendDistributor.process(uint256) (PepeFace.sol#269-293) has costly operations inside a loop:
        - currentIndex ++ (PepeFace.sol#290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
PepeFace._transferFrom(address,address,uint256) (PepeFace.sol#489-528) has a high cyclomatic complexity (13).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
DividendDistributor.REWARD (PepeFace.sol#179) is set pre-construction with a non-constant function or state variable:
        - IBEP20(WBNB)
PepeFace.totalFee (PepeFace.sol#376) is set pre-construction with a non-constant function or state variable:
        - liquidityFee.add(marketingFee).add(stakingFee).add(reflectionFee)
PepeFace.autoLiquidityReceiver (PepeFace.sol#380) is set pre-construction with a non-constant function or state variable:
        - MKT
PepeFace.marketingFeeReceiver (PepeFace.sol#381) is set pre-construction with a non-constant function or state variable:
        - MKT
PepeFace.stakingReceiver (PepeFace.sol#382) is set pre-construction with a non-constant function or state variable:
        - MKT
PepeFace.swapThreshold (PepeFace.sol#405) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.mul(10).div(10000)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Low level call in PepeFace.swapBack() (PepeFace.sol#614-677):
        - (success) = address(stakingReceiver).call{gas: 30000,value: amountBNBStaking}() (PepeFace.sol#642)
        - (success_scope_0) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}() (PepeFace.sol#650)
        - (success_scope_2) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNB_scope_1}() (PepeFace.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function IDEXRouter.WETH() (PepeFace.sol#114) is not in mixedCase
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minPeriod (PepeFace.sol#219) is not in mixedCase
Parameter DividendDistributor.setDistributionCriteria(uint256,uint256)._minDistribution (PepeFace.sol#219) is not in mixedCase
Parameter DividendDistributor.setRewardToken(address)._address (PepeFace.sol#240) is not in mixedCase
Parameter DividendDistributor.setWBNB(address)._address (PepeFace.sol#244) is not in mixedCase
Variable DividendDistributor.WBNB (PepeFace.sol#178) is not in mixedCase
Variable DividendDistributor.REWARD (PepeFace.sol#179) is not in mixedCase
Parameter PepeFace.setRewardToken(address)._rewardTokenAddress (PepeFace.sol#472) is not in mixedCase
Function PepeFace.is_buy(address,address) (PepeFace.sol#569-572) is not in mixedCase
Function PepeFace.is_sell(address,address) (PepeFace.sol#574-577) is not in mixedCase
Parameter PepeFace.getLpPair(address)._contract (PepeFace.sol#593) is not in mixedCase
Function PepeFace.EnableTrading() (PepeFace.sol#598-601) is not in mixedCase
Parameter PepeFace.recoverWrongTokens(address,uint256)._tokenAddress (PepeFace.sol#608) is not in mixedCase
Parameter PepeFace.recoverWrongTokens(address,uint256)._tokenAmount (PepeFace.sol#608) is not in mixedCase
Parameter PepeFace.setTransferBuyFee(uint256,uint256)._transferTaxRate (PepeFace.sol#689) is not in mixedCase
Parameter PepeFace.setTransferBuyFee(uint256,uint256)._buyTaxRate (PepeFace.sol#689) is not in mixedCase
Parameter PepeFace.setFeeDistribution(uint256,uint256,uint256,uint256)._liquidityFee (PepeFace.sol#698) is not in mixedCase
Parameter PepeFace.setFeeDistribution(uint256,uint256,uint256,uint256)._reflectionFee (PepeFace.sol#698) is not in mixedCase
Parameter PepeFace.setFeeDistribution(uint256,uint256,uint256,uint256)._marketingFee (PepeFace.sol#698) is not in mixedCase
Parameter PepeFace.setFeeDistribution(uint256,uint256,uint256,uint256)._stakingFee (PepeFace.sol#698) is not in mixedCase
Parameter PepeFace.changeWallets(address,address,address)._marketingReceiver (PepeFace.sol#712) is not in mixedCase
Parameter PepeFace.changeWallets(address,address,address)._stakingReceiver (PepeFace.sol#712) is not in mixedCase
Parameter PepeFace.changeWallets(address,address,address)._autoLiquidityReceiver (PepeFace.sol#712) is not in mixedCase
Parameter PepeFace.setSwapBackSettings(bool,uint256)._enabled (PepeFace.sol#721) is not in mixedCase
Parameter PepeFace.setSwapBackSettings(bool,uint256)._amount (PepeFace.sol#721) is not in mixedCase
Parameter PepeFace.setTargetLiquidity(uint256,uint256)._target (PepeFace.sol#726) is not in mixedCase
Parameter PepeFace.setTargetLiquidity(uint256,uint256)._denominator (PepeFace.sol#726) is not in mixedCase
Parameter PepeFace.setDistributionCriteria(uint256,uint256)._minPeriod (PepeFace.sol#733) is not in mixedCase
Parameter PepeFace.setDistributionCriteria(uint256,uint256)._minDistribution (PepeFace.sol#733) is not in mixedCase
Parameter PepeFace.setAutorizerAddress(address,bool)._autorizer (PepeFace.sol#747) is not in mixedCase
Variable PepeFace.REWARD (PepeFace.sol#347) is not in mixedCase
Variable PepeFace.MKT (PepeFace.sol#351) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PepeFace.sol#119) is too similar to IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PepeFace.sol#120)
Variable PepeFace.swapBack().success_scope_0 (PepeFace.sol#650) is too similar to PepeFace.swapBack().success_scope_2 (PepeFace.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
DividendDistributor.dividendsPerShareAccuracyFactor (PepeFace.sol#192) should be constant
PepeFace.MKT (PepeFace.sol#351) should be constant
PepeFace._totalSupply (PepeFace.sol#357) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
DividendDistributor._token (PepeFace.sol#178) should be immutable
DividendDistributor.router (PepeFace.sol#180) should be immutable
PepeFace.pair (PepeFace.sol#388) should be immutable
PepeFace.router (PepeFace.sol#387) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:PepeFace.sol analyzed (8 contracts with 93 detectors), 73 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

1- Approve (passed):

https://testnet.bscscan.com/tx/0x40a47fa61a50d4ac644ad464d7d5c8c2c3fb3ccee3c7e828bcc41fb1464dcb44

2- Approve Max (passed):

https://testnet.bscscan.com/tx/0x0514ebd19e75827b68b1e058878b931071434562ecb2f22315f2451131c04327

3- Change Lp Pair (passed):

https://testnet.bscscan.com/tx/0x69f694ef5eb19d54cae87ccb933660621e9f7ccef6ca21b23b80eb50df709203

4- Enable Trading (passed):

https://testnet.bscscan.com/tx/0xee08fb2ac55cefbdeca852b142e5acbba2b974a76594922a7c74ef37b047adc5

5-Clear Stuck Balance (passed):

https://testnet.bscscan.com/tx/0xab78a57eae38e002e6a4842a4abd8982cfad411cad007be581bbce90f03577b5

6- Set Autorizer Address (passed):

https://testnet.bscscan.com/tx/0x65a96f83103dcd49d3dcb46476c58ac0a71f716d8ed1432dbd0767a4b672ffdb

7- Set Target Liquidity (passed):

https://testnet.bscscan.com/tx/0xfeb494ae22b02a337ac35bf86ea13117c659629fb3b94523f38bbec589c61c2c

FUNCTIONAL TESTING

8- Set Transfer Buy Fee (passed):

https://testnet.bscscan.com/tx/0xbe1fb4a6e73317a29026acd6bc9aec96f7d3662c133496de65c33917072ab074

9- Transfer (passed):

https://testnet.bscscan.com/tx/0x63bc2287ff53b16865e2614b65a45cf40b440d4633f8fedfb4bd9f21a2f2fea9

10- Set Is Fee Exempt (passed):

https://testnet.bscscan.com/tx/0x506cc577ee6200501d4bd0b61c1e3ee8eb7cf33580c9c9190ca88b7e4d264db6

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
    require(!swapTokenEnabled, "Cannot re-enable trading");
    swapTokenEnabled = true;
 }
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Visibility

Severity: Low

subject: Missing Visibility

Status: Open

Overview:

It's simply saying that no visibility was specified, so it's going with the default. This has been related to security issues in contracts.

```
address _token;

address WBNB =
0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
  IBEP20 REWARD = IBEP20(WBNB);
  IDEXRouter router;

mapping (address => uint256) shareholderIndexes;
mapping (address => uint256) shareholderClaims;

 uint256 currentIndex;
 bool initialized;
```

Suggestion:

You can easily silence the warning by adding the visibility public/private.

MANUAL TESTING

Optimization

Severity: Informational

subject: Remove Safe Math

Status: Open

Line: 9-46

Overview:

compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow, It is recommended to remove the unwanted code in order to avoid high gas fees.

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