auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PEPE MUSK

DATED : 18 May 23'

AUDIT SUMMARY

Project name – PEPE MUSK

Date: 18 May, 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	3
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/0xd532c35c88b6bbccba323e1614505587f21441a2

Token Information

Token Name : PEPE MUSK

Token Symbol: PPM

Decimals: 18

Token Supply: 21,000,000,000

Token Address:
0x1619bcc91e2c90E0eDfb2C95EC897f0C0F393541

Checksum:
aa59b6777bc026949a1b49f5f48ca0b238c1b721

Owner:
0x25D1A986b645e5cDF5b8c5893972e68F14018FCa

Deployer:
0x25D1A986b645e5cDF5b8c5893972e68F14018FCa

TOKEN OVERVIEW

Fees:

Buy Fees: 2%

Sell Fees: 2%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: 0x25D1A986b645e5cDF5b8c5893972e68F14018FCa

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: enabling trades

- excluding from fees
- including in fees
- changing swap threshold

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

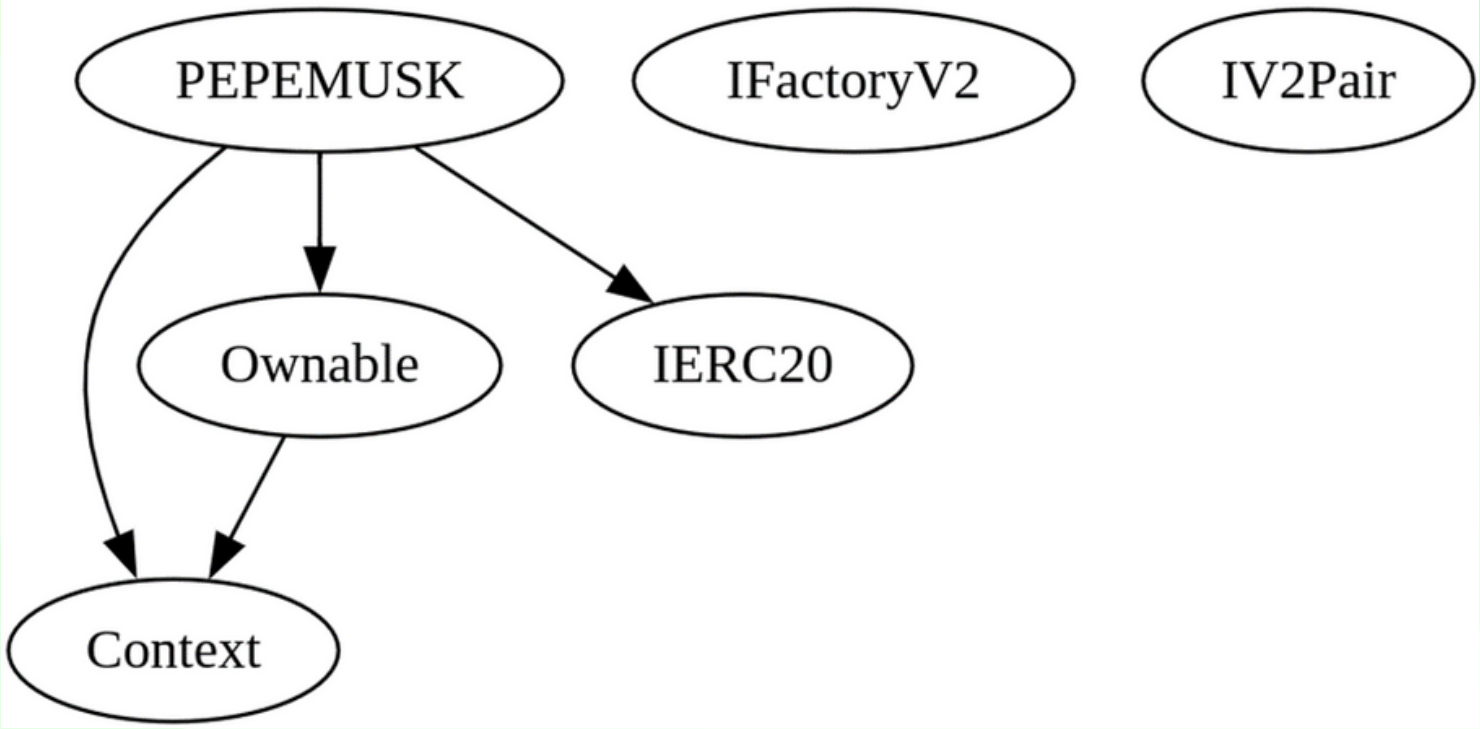
CLASSIFICATION OF RISK

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	0
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	3

INHERITANCE TREE



FINDINGS TABLE

Category	Subject	Severity	Status	Suggestion Summary
Centralization	Trading restrictions	Medium	Open	Consider community voting or a predetermined time for enabling trading
Centralization	Owner privileges	Minor/Informational	Not applicable	Implement decentralized governance for changes to contract's parameters and settings
Centralization	Fee setting	Minor/Informational	Open	Implement a function to update buy/sell/transfer fees based on market conditions within a 0-25% range
Gas optimization	Incorrect revert condition in totalSupply and decimals functions	Minor	Open	Remove the unnecessary revert condition in the totalSupply() and decimals() functions

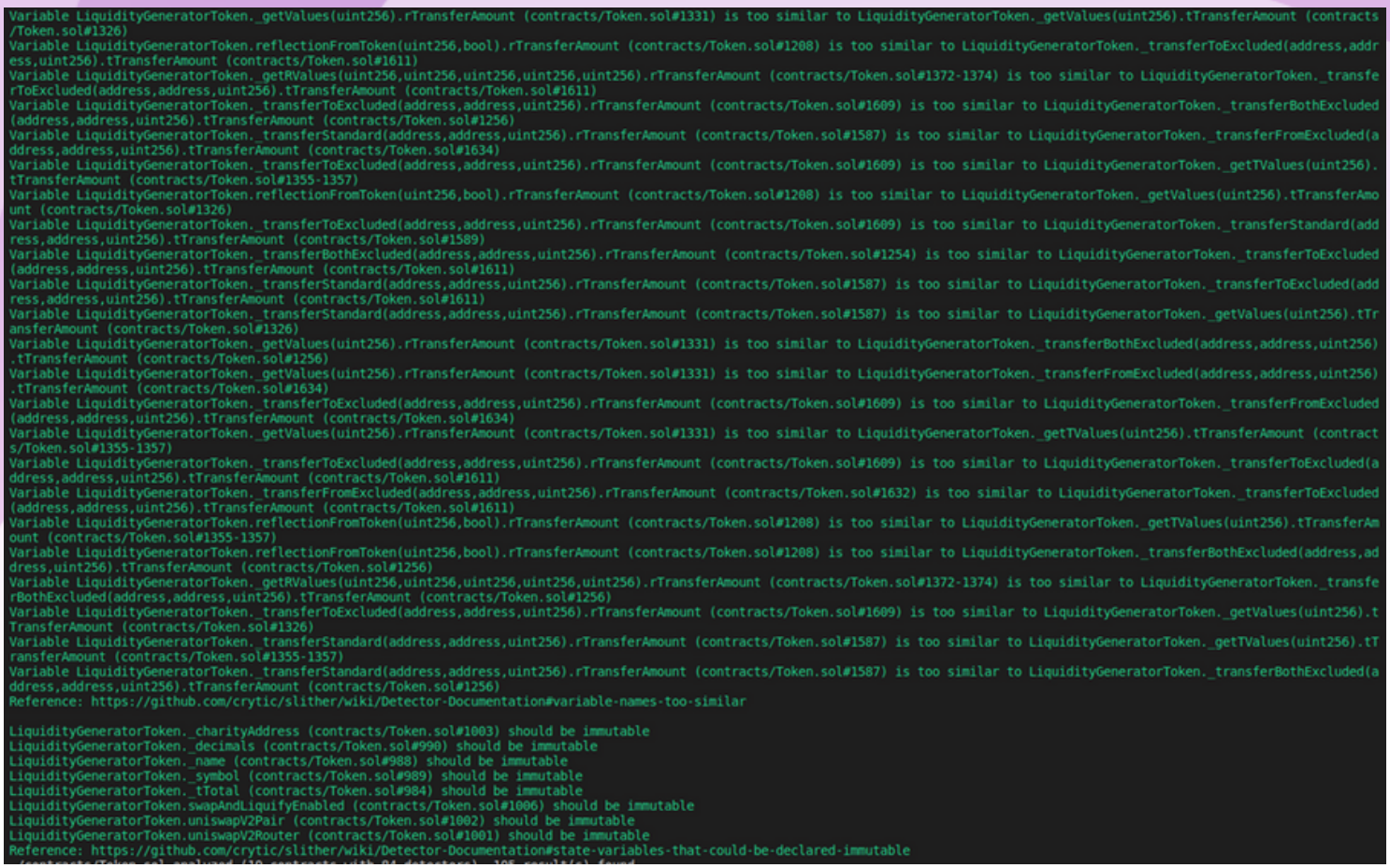
CONTRACT ASSESSMENT

Function Name	Description	Risks
transfer	Transfers tokens from the sender to the recipient.	If the recipient address is invalid or the sender has insufficient balance, the transfer will fail.
approve	Approves the spender to spend a certain amount of tokens on behalf of the owner.	If the spender address is invalid, the approval will fail.
transferFrom	Transfers tokens from one address to another, using the allowance mechanism.	If the sender or recipient address is invalid, or the sender has insufficient allowance, the transfer will fail.
setNoFeeWallet	Sets whether a wallet should be exempt from fees.	Only callable by the owner. If used maliciously, it could exempt certain addresses from fees.
changeLpPair	Changes the liquidity pool pair.	Only callable by the owner. If used maliciously, it could change the liquidity pool pair to an invalid address.
toggleCanSwapFees	Toggles whether fees can be swapped.	Only callable by the owner. If used maliciously, it could disable the swapping of fees.
changeWallets	Changes the marketing wallet address.	Only callable by the owner. If used maliciously, it could change the marketing wallet to an invalid address.
setPresaleAddress	Sets whether an address is a presale address.	Only callable by the owner. If used maliciously, it could exempt certain addresses from fees and restrictions.
enableTrading	Enables trading of the token.	Only callable by the owner. If used maliciously, it could enable trading before the presale is complete.

POINTS TO NOTE

- Owner is not able to change current fees (2% for buy/sell and 0% for transfers)
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to limit buy/sell/transfer/wallet amounts
- Owner must enable trades manually
- Owner is not able to mint new tokens

STATIC ANALYSIS



```
Variable LiquidityGeneratorToken._getValues(uint256).rTransferAmount (contracts/Token.sol#1331) is too similar to LiquidityGeneratorToken._getValues(uint256).tTransferAmount (contracts
/Token.sol#1326)
Variable LiquidityGeneratorToken.reflectionFromToken(uint256,bool).rTransferAmount (contracts/Token.sol#1208) is too similar to LiquidityGeneratorToken._transferToExcluded(address,addr
ess,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1372-1374) is too similar to LiquidityGeneratorToken._transfe
rToExcluded(address,address,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._transferBothExcluded
(address,address,uint256).tTransferAmount (contracts/Token.sol#1256)
Variable LiquidityGeneratorToken._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#1587) is too similar to LiquidityGeneratorToken._transferFromExcluded(a
ddress,address,uint256).tTransferAmount (contracts/Token.sol#1634)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._getTValues(uint256).
tTransferAmount (contracts/Token.sol#1355-1357)
Variable LiquidityGeneratorToken.reflectionFromToken(uint256,bool).rTransferAmount (contracts/Token.sol#1208) is too similar to LiquidityGeneratorToken._getValues(uint256).tTransferAmo
unt (contracts/Token.sol#1326)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._transferStandard(add
ress,address,uint256).tTransferAmount (contracts/Token.sol#1589)
Variable LiquidityGeneratorToken._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1254) is too similar to LiquidityGeneratorToken._transferToExcluded
(address,address,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#1587) is too similar to LiquidityGeneratorToken._transferToExcluded(add
ress,address,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#1587) is too similar to LiquidityGeneratorToken._getValues(uint256).tTr
ansferAmount (contracts/Token.sol#1326)
Variable LiquidityGeneratorToken._getValues(uint256).rTransferAmount (contracts/Token.sol#1331) is too similar to LiquidityGeneratorToken._transferBothExcluded(address,address,uint256)
.tTransferAmount (contracts/Token.sol#1256)
Variable LiquidityGeneratorToken._getValues(uint256).rTransferAmount (contracts/Token.sol#1331) is too similar to LiquidityGeneratorToken._transferFromExcluded(address,address,uint256)
.tTransferAmount (contracts/Token.sol#1634)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._transferFromExcluded
(address,address,uint256).tTransferAmount (contracts/Token.sol#1634)
Variable LiquidityGeneratorToken._getValues(uint256).rTransferAmount (contracts/Token.sol#1331) is too similar to LiquidityGeneratorToken._getTValues(uint256).tTransferAmount (contract
s/Token.sol#1355-1357)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._transferToExcluded(a
ddress,address,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1632) is too similar to LiquidityGeneratorToken._transferToExcluded
(address,address,uint256).tTransferAmount (contracts/Token.sol#1611)
Variable LiquidityGeneratorToken.reflectionFromToken(uint256,bool).rTransferAmount (contracts/Token.sol#1208) is too similar to LiquidityGeneratorToken._getTValues(uint256).tTransferAm
ount (contracts/Token.sol#1355-1357)
Variable LiquidityGeneratorToken.reflectionFromToken(uint256,bool).rTransferAmount (contracts/Token.sol#1208) is too similar to LiquidityGeneratorToken._transferBothExcluded(address,ad
dress,uint256).tTransferAmount (contracts/Token.sol#1256)
Variable LiquidityGeneratorToken._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#1372-1374) is too similar to LiquidityGeneratorToken._transfe
rBothExcluded(address,address,uint256).tTransferAmount (contracts/Token.sol#1256)
Variable LiquidityGeneratorToken._transferToExcluded(address,address,uint256).rTransferAmount (contracts/Token.sol#1609) is too similar to LiquidityGeneratorToken._getValues(uint256).t
TransferAmount (contracts/Token.sol#1326)
Variable LiquidityGeneratorToken._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#1587) is too similar to LiquidityGeneratorToken._getTValues(uint256).tT
ransferAmount (contracts/Token.sol#1355-1357)
Variable LiquidityGeneratorToken._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#1587) is too similar to LiquidityGeneratorToken._transferBothExcluded(a
ddress,address,uint256).tTransferAmount (contracts/Token.sol#1256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

LiquidityGeneratorToken._charityAddress (contracts/Token.sol#1003) should be immutable
LiquidityGeneratorToken._decimals (contracts/Token.sol#990) should be immutable
LiquidityGeneratorToken._name (contracts/Token.sol#988) should be immutable
LiquidityGeneratorToken._symbol (contracts/Token.sol#989) should be immutable
LiquidityGeneratorToken._tTotal (contracts/Token.sol#984) should be immutable
LiquidityGeneratorToken.swapAndLiquifyEnabled (contracts/Token.sol#1006) should be immutable
LiquidityGeneratorToken.uniswapV2Pair (contracts/Token.sol#1002) should be immutable
LiquidityGeneratorToken.uniswapV2Router (contracts/Token.sol#1001) 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

Router (PCS V2):

0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x52fe289fe0af726bbab7eab354f30f02eb82e3761b6a54972ffc41e2e80226b6

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x4d6b58d7506607d0d153933178502903951 13af750dc4decaf70373b3799a14a

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5f80aa037b1fcfbaceeab0df910187e9c7b4f278589d6262e62f0f9eb0350375

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd4d1084634a586343c336df5b07ddfac7fd69e64930c7445ba5ae42b76e2dc6c

5- Buying from a regular wallet (2% tax) (passed):
https://testnet.bscscan.com/tx/0x104bce6f2dc791ae7bebcd567a8998684ab46d04277ade10cc7c896567947e86

6- Selling from a regular wallet (2% tax) (passed):
https://testnet.bscscan.com/tx/0x4e31ab91436e80945ea4c948b29d6475fab9db50da91581b1db5c3faa22d570d

FUNCTIONAL TESTING

7- Transferring from a regular wallet (0% tax) (passed):
https://testnet.bscscan.com/tx/0x9e709491793caf71b75cca5344d0b2977e0263ec1807d1009a083b2b8aa4112a

8- Internal swap (marketing bnb) (passed):
https://testnet.bscscan.com/tx/0x4e31ab91436e80945ea4c948b29d6475fab9db50da91581b1db5c3faa22d570d

MANUAL TESTING

Category: **Centralization**
Subject: Trading restrictions
Severity: Medium
Status: Not applicable
Overview:
The contract has a function to enable trading, which can only be called by the owner. This introduces a centralization risk as the owner has control over when trading can begin.

Code:
- enableTrading() (line 239)

Suggestion:
Consider allowing the community to vote on when trading should be enabled, or set a predetermined time for trading to begin. This can help reduce the centralization risk and ensure that no single entity has control over the start of trading.

,,

MANUAL TESTING

Category: Centralization
Subject: Owner privileges
Severity: Minor / Informational
Status: Not applicable
Overview:
The contract has several functions that can only be called by the owner, which introduces centralization risks. These functions include setting no-fee wallets, changing the liquidity pair address (which can affect taxes), toggling internal swaps, changing wallets, setting presale addresses, and enabling trading.

Code:
- setNoFeeWallet(address account, bool enabled) (line 146)
- changeLpPair(address newPair) (line 179)
- toggleCanSwapFees(bool yesno) (line 184)
- changeWallets(address marketing) (line 213)
- setPresaleAddress(address presale, bool yesno) (line 233)
- enableTrading() (line 239)

Suggestion:
To mitigate centralization risks, consider implementing a decentralized governance mechanism that allows the community to vote on changes to the contract's parameters and settings. This can help ensure that no single entity has control over the contract's functionality and reduces the risk of centralization.

""

MANUAL TESTING

Category: **Centralization**
Subject: **Fee setting**
Severity: Minor / Informational
Stauts: Open
Overview:
The contract has hardcoded fees for buying, selling, and transferring tokens. These fees are not adjustable by the owner. Its suggested to have a function for adjusting fees in different market conditions.

Code:
- uint256 public buyfee = 20;
- uint256 public sellfee = 20;
- uint256 public transferfee = 0;

Suggestion:
Consider implementing a function to be able to update buy/sell/transfer fees based on different market condition withing a reasonable range (0-25% total fees)

MANUAL TESTING

Category: Gas optimization
Subject: Incorrect revert condition in totalSupply and decimals functions
Severity: Minor
Overview:
The totalSupply() and decimals() functions have a revert condition that checks if _totalSupply is equal to 0. However, this condition will never be true since _totalSupply is a constant value of 21_000_000_000 * 10 ** 18. This makes the revert condition unnecessary and can be removed.

Redundant code:

```
function totalSupply() external pure override returns (uint256) {
    if (_totalSupply == 0) {
        revert();
    }
    return _totalSupply;
}

function decimals() external pure override returns (uint8) {
    if (_totalSupply == 0) {
        revert();
    }
    return _decimals;
}
```

Suggestion:
Remove the unnecessary revert condition in the totalSupply() and decimals() functions.

Updated code:

```
function totalSupply() external pure override returns (uint256) {
    return _totalSupply;
}

function decimals() external pure override returns (uint8) {
    return _decimals;
}
```

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