AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PEPEBULL

DATED : 2 March, 2024

AUDIT SUMMARY

Project name – PEPEBULL

Date: 2 March, 2024

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	1	2
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/0x72e19b9748d5b4f6532fdf62dd84038040a159d6#code

Token Information

Token Name : PEPEBULL

Token Symbol: PEPEBULL

Decimals: 18

Total Supply: 1000000000

Network: Binance smart chain

Token Type: BEP-20

Token Address:
0x9B6F23c9bfe27dfb41e6025f39e7303631Cd0d27

Checksum:
B67acbefe2a12642d388659dffd20713

Owner:
0xC30555dF47CF9A7e8EAa486CF19001D4950569E5
(at time of writing the audit)

Deployer:
0xC30555dF47CF9A7e8EAa486CF19001D4950569E5

TOKEN OVERVIEW

Fees:
BuyBack Tax: 3%
Sell Tax: 3%
Transfer Tax: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

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.

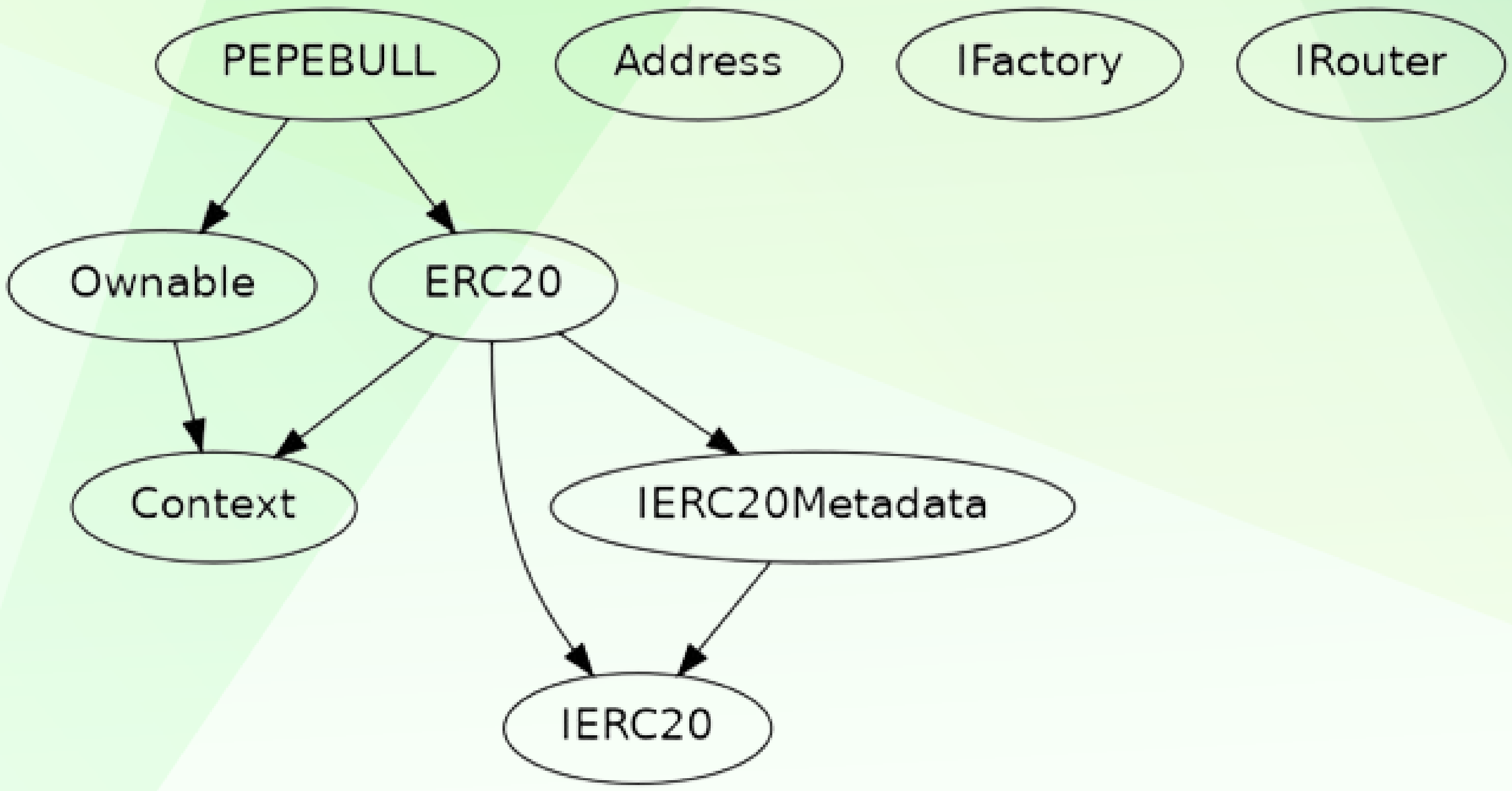
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

auditace.tech

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

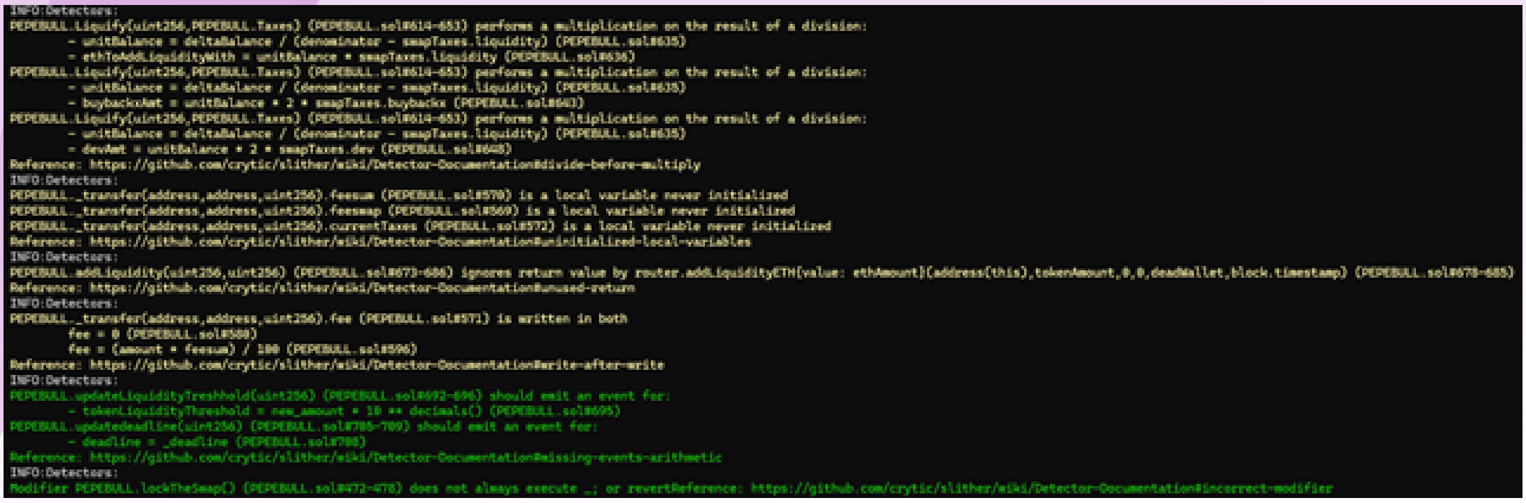
INHERITANCE TREE



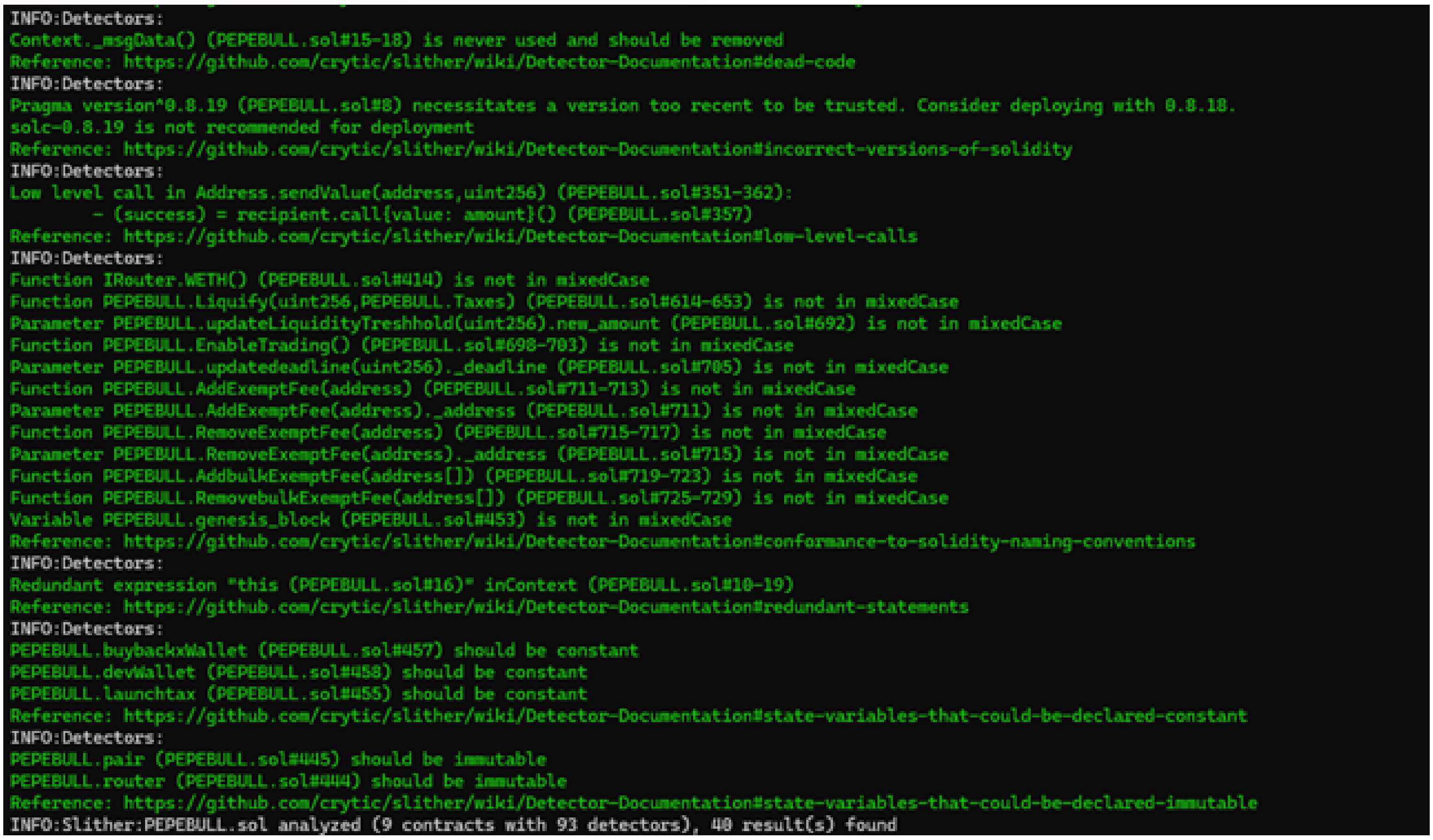
STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.



```
INFO:Detectors:
PEPEBULL.Liquify(uint256,PEPEBULL.Taxes) (PEPEBULL.sol#614-653) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (PEPEBULL.sol#635)
        - ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity (PEPEBULL.sol#636)
PEPEBULL.Liquify(uint256,PEPEBULL.Taxes) (PEPEBULL.sol#614-653) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (PEPEBULL.sol#635)
        - buybackAmt = unitBalance * 2 * swapTaxes.buyback (PEPEBULL.sol#643)
PEPEBULL.Liquify(uint256,PEPEBULL.Taxes) (PEPEBULL.sol#614-653) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (PEPEBULL.sol#635)
        - devAmt = unitBalance * 2 * swapTaxes.dev (PEPEBULL.sol#648)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
PEPEBULL._transfer(address,address,uint256).feesum (PEPEBULL.sol#570) is a local variable never initialized
PEPEBULL._transfer(address,address,uint256).feeswap (PEPEBULL.sol#569) is a local variable never initialized
PEPEBULL._transfer(address,address,uint256).currentTaxes (PEPEBULL.sol#572) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PEPEBULL.addLiquidity(uint256,uint256) (PEPEBULL.sol#673-686) ignores return value by router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (PEPEBULL.sol#678-685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
PEPEBULL._transfer(address,address,uint256).fee (PEPEBULL.sol#571) is written in both
        fee = 0 (PEPEBULL.sol#580)
        fee = (amount * feesum) / 100 (PEPEBULL.sol#596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
INFO:Detectors:
PEPEBULL.updateLiquidityTreshhold(uint256) (PEPEBULL.sol#692-696) should emit an event for:
        - tokenLiquidityThreshold = new_amount * 10 ** decimals() (PEPEBULL.sol#695)
PEPEBULL.updatedeadline(uint256) (PEPEBULL.sol#705-709) should emit an event for:
        - deadline = _deadline (PEPEBULL.sol#708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Modifier PEPEBULL.lockTheSwap() (PEPEBULL.sol#472-478) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
```



```
INFO:Detectors:
Context._msgData() (PEPEBULL.sol#15-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (PEPEBULL.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (PEPEBULL.sol#351-362):
        - (success) = recipient.call{value: amount}() (PEPEBULL.sol#357)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter.WETH() (PEPEBULL.sol#414) is not in mixedCase
Function PEPEBULL.Liquify(uint256,PEPEBULL.Taxes) (PEPEBULL.sol#614-653) is not in mixedCase
Parameter PEPEBULL.updateLiquidityTreshhold(uint256).new_amount (PEPEBULL.sol#692) is not in mixedCase
Function PEPEBULL.EnableTrading() (PEPEBULL.sol#698-703) is not in mixedCase
Parameter PEPEBULL.updatedeadline(uint256)._deadline (PEPEBULL.sol#705) is not in mixedCase
Function PEPEBULL.AddExemptFee(address) (PEPEBULL.sol#711-713) is not in mixedCase
Parameter PEPEBULL.AddExemptFee(address)._address (PEPEBULL.sol#711) is not in mixedCase
Function PEPEBULL.RemoveExemptFee(address) (PEPEBULL.sol#715-717) is not in mixedCase
Parameter PEPEBULL.RemoveExemptFee(address)._address (PEPEBULL.sol#715) is not in mixedCase
Function PEPEBULL.AddbulkExemptFee(address[]) (PEPEBULL.sol#719-723) is not in mixedCase
Function PEPEBULL.RemovebulkExemptFee(address[]) (PEPEBULL.sol#725-729) is not in mixedCase
Variable PEPEBULL.genesis_block (PEPEBULL.sol#453) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (PEPEBULL.sol#16)" inContext (PEPEBULL.sol#10-19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
PEPEBULL.buybackWallet (PEPEBULL.sol#457) should be constant
PEPEBULL.devWallet (PEPEBULL.sol#458) should be constant
PEPEBULL.launchtax (PEPEBULL.sol#455) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
PEPEBULL.pair (PEPEBULL.sol#445) should be immutable
PEPEBULL.router (PEPEBULL.sol#444) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:PEPEBULL.sol analyzed (9 contracts with 93 detectors), 40 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):
https://testnet.bscscan.com/tx/0x8b69d6abc3b630da2b847e55b72435f6a5e4fc8670ad94f854e013f1f1be1435

2- Add Exempt Fee (passed):
https://testnet.bscscan.com/tx/0x1b2255736d724de9b6299730b3fd77ce6b4a22aa81af9c24a5194beb613bd4ed

3- Remove Exempt Fee (passed):
https://testnet.bscscan.com/tx/0x7bbd70b3dfae44cd1d23f20ac1a6b4751941724e48973bda083b0ff5ab362ab4

4- Addbulk Exempt Fee (passed):
https://testnet.bscscan.com/tx/0x4823342473af258d2c8a4210987e59af4e091d4eb1454838d13c728a34ed10b1

5- Removebulk Exempt Fee (passed):
https://testnet.bscscan.com/tx/0xe9ffcccf5c075a5633b19f4ea14ce0648feb4c217c99a7df44da6bf74516bb80

6- Update Liquidity Provide (passed):
https://testnet.bscscan.com/tx/0xf87cff294c92259228d603f85b43c3a0e0d9a8e5d5939836ffe36e2417e4a7f8

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can update Liquidity threshold.
- The owner can Enable trading.
- The owner can update the deadline.
- The owner can add/remove address from exempt fees.
- The owner can rescue BEP20.

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.

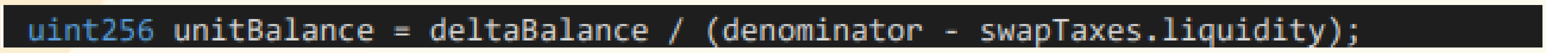
Findings

Severity	Found
🟥 **Critical**	0
🟫 **High-Risk**	0
🟧 **Medium-Risk**	1
🟨 **Low-Risk**	1
🟦 **Gas Optimization / Suggestions**	2

MANUAL TESTING

Centralization: Divide before multiply
Severity: Medium
Status: Open

Overview:
Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.



```
uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
```

Suggestion:
Consider ordering multiplication before division.

MANUAL TESTING

Centralization – Missing Events

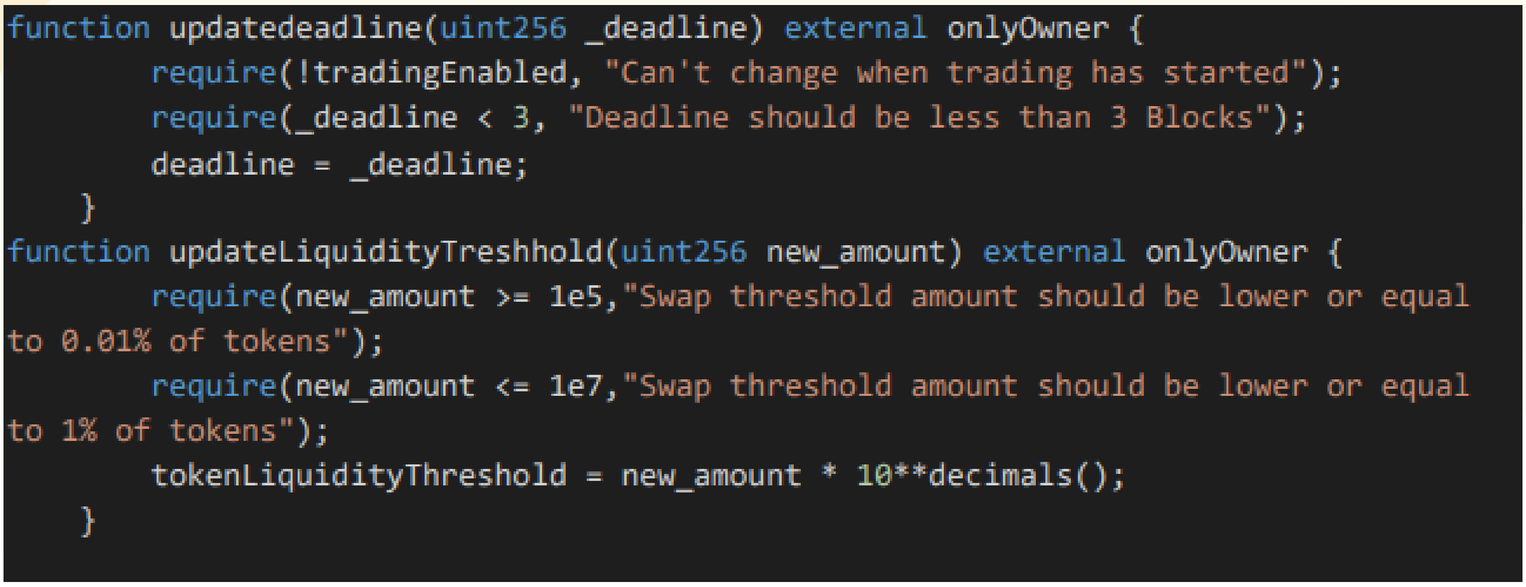
Severity: Low

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 updatedeadline(uint256 _deadline) external onlyOwner {
        require(!tradingEnabled, "Can't change when trading has started");
        require(_deadline < 3, "Deadline should be less than 3 Blocks");
        deadline = _deadline;
    }
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
        require(new_amount >= 1e5,"Swap threshold amount should be lower or equal
to 0.01% of tokens");
        require(new_amount <= 1e7,"Swap threshold amount should be lower or equal
to 1% of tokens");
        tokenLiquidityThreshold = new_amount * 10**decimals();
    }
```

MANUAL TESTING

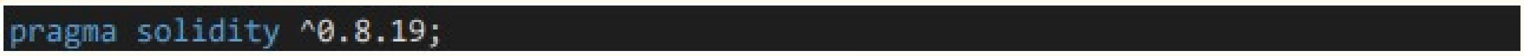
Optimization

Severity: Informational

Subject: Floating Pragma 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 ^0.8.19;
```

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

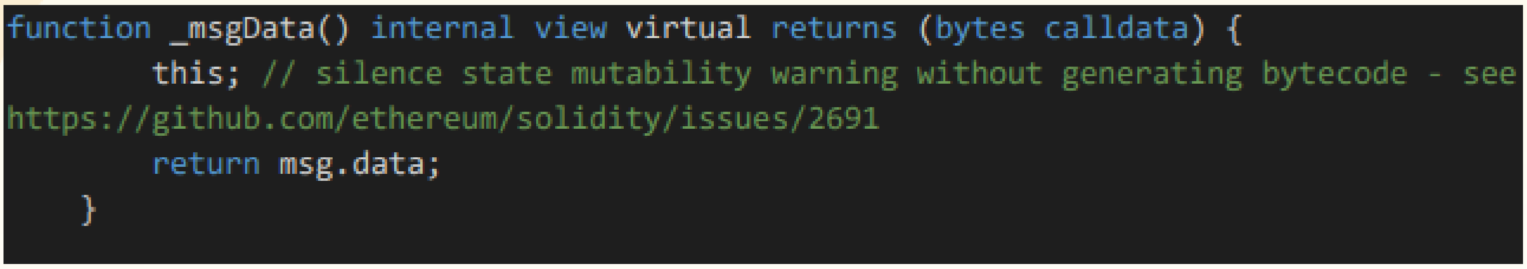
Severity: Optimization

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