auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Starlink AI

DATED : 02 June 24'

MANUAL TESTING

Centralization – Buy and Sell Fees.

Severity: High

Function: setBuyFees, setSellFees

Status: FIXED (Ownership Renounced)

Overview:

The owner can set the buy and sell fees up to 40%, which is not recommended.

```
function setBuyFees(
    uint256 _marketingFee,
    uint256 _liquidityFee,
    uint256 _DevelopmentFee
  ) external onlyOwner {
    buyMarketingFee = _marketingFee;
    buyLPFee = _liquidityFee;
    buyDevelopmentFee = _DevelopmentFee;
    totalBuyFee = buyMarketingFee + buyLPFee + buyDevelopmentFee;
    require(totalBuyFee <= 40, "Total buy fee cannot be higher than
40%");
    emit BuyFeeUpdated(totalBuyFee, buyMarketingFee, buyLPFee,
buyDevelopmentFee);
  }
function setSellFees(
    uint256 _marketingFee,
    uint256 _liquidityFee,
    uint256 _DevelopmentFee
  ) external onlyOwner {
    sellMarketingFee = _marketingFee;
    sellLpFee = _liquidityFee;
    sellDevelopmentFee = _DevelopmentFee;
    totalSellFee = sellMarketingFee + sellLpFee + sellDevelopmentFee;
```

MANUAL TESTING

```
    require(
      totalSellFee <= 40,
      "Total sell fee cannot be higher than 40%"
    );
    emit SellFeeUpdated(totalSellFee, sellMarketingFee, sellLpFee,
sellDevelopmentFee);
```

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
Function: OpenTrading
Status: FIXED (Ownership Renounced)

Overview:
The OpenTrading 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 openTrading() external onlyOwner {
    tradingOn = true;
    swapbackEnabled = true;
    emit TradingEnabled(block.timestamp);
  }
```

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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Require Check.
Severity: High
Function: setmarketingWallet/setLPWallet/setDevWallet
Status: FIXED (Ownership Renounced)

Overview:
The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setMarketingWallet(address newWallet) external onlyOwner {
    emit MarketingReceiverUpdated(newWallet, MarketingReceiver);
    MarketingReceiver = newWallet;
  }
function setLPWallet(address newWallet) external onlyOwner {
    emit lpReceiverUpdated(newWallet, autoLPReceiver);
    autoLPReceiver = newWallet;
  }
function setDevWallet(address newWallet) external onlyOwner {
    emit devReceiverUpdated(newWallet, devReceiver);
    devReceiver = newWallet;
  }
```

Suggestion:
It is recommended that the address should not be able to be set as a contract address.

AUDIT SUMMARY

Project name – Starlink AI

Date: 02 June, 2024

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

Audit Status: **PASSED & HIGH RISK ARE FIXED**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	1	1	2
Acknowledged	0	0	0	0	0
Resolved	0	3	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/0x00dbc75b9190f7cd3e09fd338dd8e26f104eb2e6#code

Token Information

Token Address:
0x454bFf91E0543bD6e9f85AA8D0b40B9C65a2ff0F

Name: Starlink AI

Symbol: STAI

Decimals: 18

Network: EtherScan

Token Type: ERC-20

Owner: 0x0000000000000000000000000000000000000000

Deployer:
0x09Cc746fBD4B8d85bD4bd2EAb9686E7889abA5a7

Token Supply: 100000000

Checksum: A17acbefe2a12642d388659dffd20221

Testnet:
https://testnet.bscscan.com/address/0x00dbc75b9190f7cd3e09fd338dd8e26f104eb2e6#code

TOKEN OVERVIEW

Buy Fee: 5%

Sell Fee: 5%

Transfer Fee: 5%

Fee Privilege: Owner

Ownership: Renounced

Minting: None

Max Tx: No

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

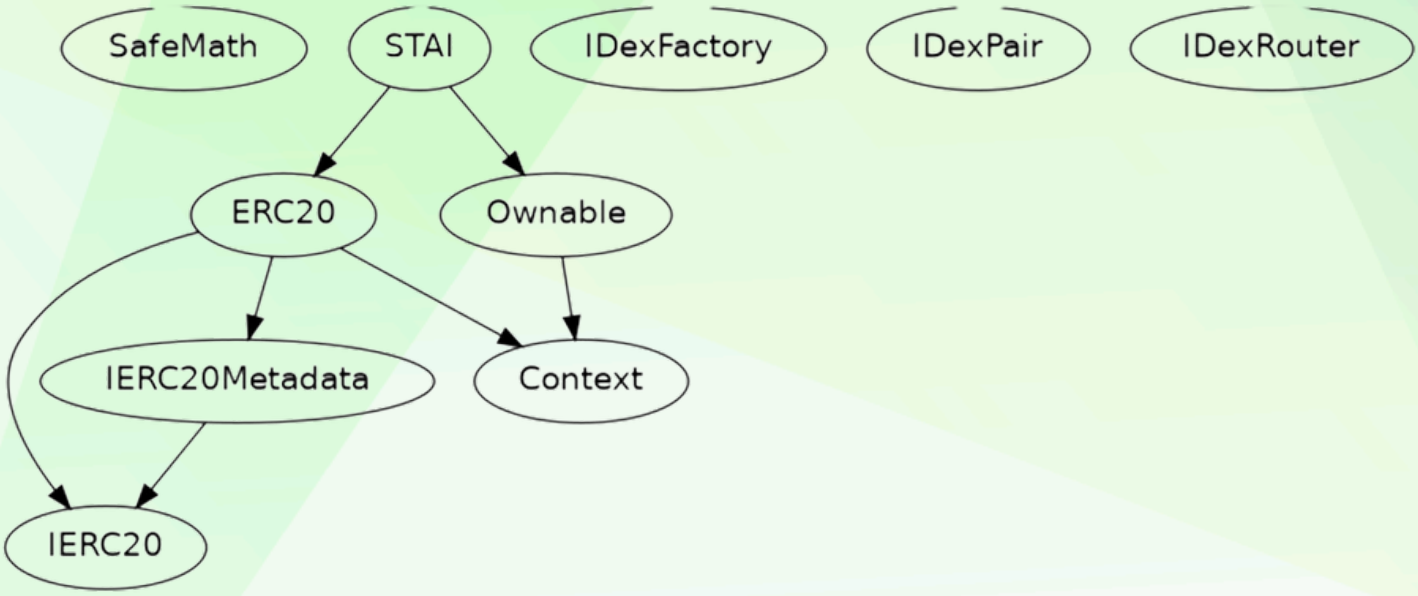
CLASSIFICATION OF RISK

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	3
Medium-Risk	1
Low-Risk	1
Gas Optimization / Suggestions	2

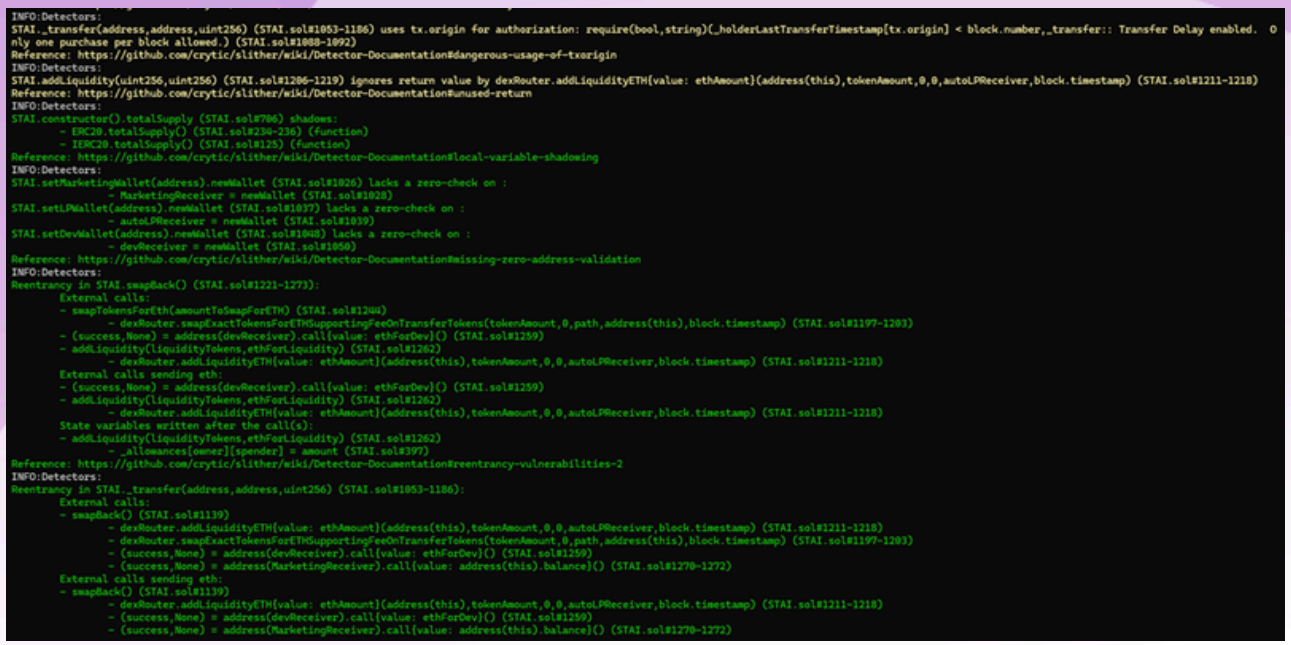
INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can set a buy/sell fee to more than 40%.
- The owner can set swap Back Settings.
- The owner can set a wallet limit of not more than 0.5%.
- The owner set up an Automated market maker pair.
- The owner can set LPwallet/DevWallet/MarketingWallet.

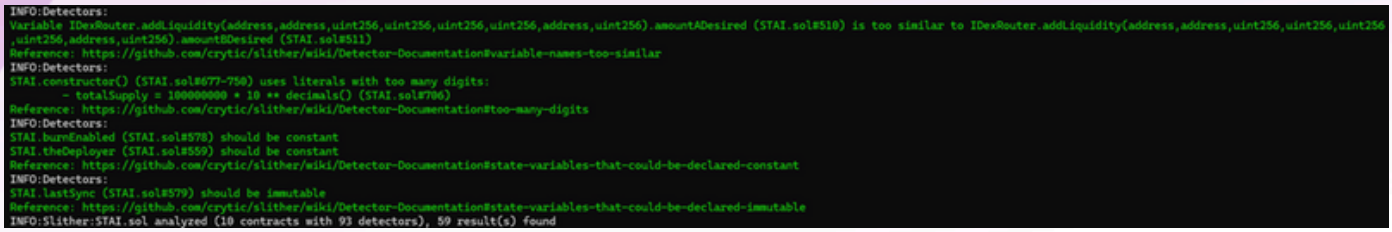
STATIC ANALYSIS



```
INFO:Detectors:
STAI._transfer(address,address,uint256) (STAI.sol#1053-1186) uses tx.origin for authorization: require(bool,string)(_holderLastTransferTimestamp[tx.origin] < block.number,_transfer:: Transfer Delay enabled.  O
nly one purchase per block allowed.) (STAI.sol#1088-1092)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin
INFO:Detectors:
STAI.addLiquidity(uint256,uint256) (STAI.sol#1206-1219) ignores return value by dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,autoLPReceiver,block.timestamp) (STAI.sol#1211-1218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
STAI.constructor().totalSupply (STAI.sol#706) shadows:
        - ERC20.totalSupply() (STAI.sol#230-236) (function)
        - IERC20.totalSupply() (STAI.sol#125) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
STAI.setMarketingWallet(address).newWallet (STAI.sol#1026) lacks a zero-check on :
                - MarketingReceiver = newWallet (STAI.sol#1028)
STAI.setLPWallet(address).newWallet (STAI.sol#1037) lacks a zero-check on :
                - autoLPReceiver = newWallet (STAI.sol#1039)
STAI.setDevWallet(address).newWallet (STAI.sol#1048) lacks a zero-check on :
                - devReceiver = newWallet (STAI.sol#1050)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in STAI.swapBack() (STAI.sol#1221-1273):
        External calls:
        - swapTokensForEth(amountToSwapForETH) (STAI.sol#1244)
                - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (STAI.sol#1197-1203)
        - (success,None) = address(devReceiver).call{value: ethForDev}() (STAI.sol#1259)
        - addLiquidity(liquidityTokens,ethForLiquidity) (STAI.sol#1262)
                - dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,autoLPReceiver,block.timestamp) (STAI.sol#1211-1218)
        External calls sending eth:
        - (success,None) = address(devReceiver).call{value: ethForDev}() (STAI.sol#1259)
        - addLiquidity(liquidityTokens,ethForLiquidity) (STAI.sol#1262)
                - dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,autoLPReceiver,block.timestamp) (STAI.sol#1211-1218)
        State variables written after the call(s):
        - addLiquidity(liquidityTokens,ethForLiquidity) (STAI.sol#1262)
                - _allowances[owner][spender] = amount (STAI.sol#397)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in STAI._transfer(address,address,uint256) (STAI.sol#1053-1186):
        External calls:
        - swapBack() (STAI.sol#1139)
                - dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,autoLPReceiver,block.timestamp) (STAI.sol#1211-1218)
                - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (STAI.sol#1197-1203)
                - (success,None) = address(devReceiver).call{value: ethForDev}() (STAI.sol#1259)
                - (success,None) = address(MarketingReceiver).call{value: address(this).balance}() (STAI.sol#1270-1272)
        External calls sending eth:
        - swapBack() (STAI.sol#1139)
                - dexRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,autoLPReceiver,block.timestamp) (STAI.sol#1211-1218)
                - (success,None) = address(devReceiver).call{value: ethForDev}() (STAI.sol#1259)
                - (success,None) = address(MarketingReceiver).call{value: address(this).balance}() (STAI.sol#1270-1272)
```

```
INFO:Detectors:
STAI._transfer(address,address,uint256) (STAI.sol#1053-1186) has a high cyclomatic complexity (16).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Context._msgData() (STAI.sol#192-194) is never used and should be removed
ERC20._burn(address,uint256) (STAI.sol#372-387) is never used and should be removed
SafeMath.add(uint256,uint256) (STAI.sol#67-69) is never used and should be removed
SafeMath.div(uint256,uint256,string) (STAI.sol#98-107) is never used and should be removed
SafeMath.mod(uint256,uint256) (STAI.sol#83-85) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (STAI.sol#109-118) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (STAI.sol#87-96) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (STAI.sol#11-20) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (STAI.sol#47-55) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (STAI.sol#57-65) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (STAI.sol#32-45) is never used and should be removed
SafeMath.trySub(uint256,uint256) (STAI.sol#22-30) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.19 (STAI.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 STAI.swapBack() (STAI.sol#1221-1273):
        - (success,None) = address(devReceiver).call{value: ethForDev}() (STAI.sol#1259)
        - (success,None) = address(MarketingReceiver).call{value: address(this).balance}() (STAI.sol#1270-1272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IDexRouter.WETH() (STAI.sol#505) is not in mixedCase
Event STAI.devReceiverUpdated(address,address) (STAI.sol#640-643) is not in CapWords
Event STAI.lpReceiverUpdated(address,address) (STAI.sol#645-648) is not in CapWords
Parameter STAI.checkAddressPermissions(address)._target (STAI.sol#844) is not in mixedCase
Function STAI.RemoveLimits() (STAI.sol#881-884) is not in mixedCase
Parameter STAI.setSwapBackSettings(bool,uint256,uint256)._enabled (STAI.sol#892) is not in mixedCase
Parameter STAI.setSwapBackSettings(bool,uint256,uint256)._min (STAI.sol#893) is not in mixedCase
Parameter STAI.setSwapBackSettings(bool,uint256,uint256)._max (STAI.sol#894) is not in mixedCase
Parameter STAI.setBuyFees(uint256,uint256,uint256)._marketingFee (STAI.sol#948) is not in mixedCase
Parameter STAI.setBuyFees(uint256,uint256,uint256)._liquidityFee (STAI.sol#949) is not in mixedCase
Parameter STAI.setBuyFees(uint256,uint256,uint256)._DevelopmentFee (STAI.sol#950) is not in mixedCase
Parameter STAI.setSellFees(uint256,uint256,uint256)._marketingFee (STAI.sol#961) is not in mixedCase
Parameter STAI.setSellFees(uint256,uint256,uint256)._liquidityFee (STAI.sol#962) is not in mixedCase
Parameter STAI.setSellFees(uint256,uint256,uint256)._DevelopmentFee (STAI.sol#963) is not in mixedCase
Parameter STAI.setTransferFees(uint256,uint256,uint256)._marketingFee (STAI.sol#977) is not in mixedCase
Parameter STAI.setTransferFees(uint256,uint256,uint256)._liquidityFee (STAI.sol#978) is not in mixedCase
Parameter STAI.setTransferFees(uint256,uint256,uint256)._DevelopmentFee (STAI.sol#979) is not in mixedCase
Variable STAI.MarketingReceiver (STAI.sol#583) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

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- Open Trading (passed):

https://testnet.bscscan.com/tx/0x2e0491ca76bec774b8404f513102bda6f74c5023ce5ca3848bc9fa2601eae68c

2- Set Marketing Wallet (passed):

https://testnet.bscscan.com/tx/0x198e41a239181a55a65f454c08540391ddbc0cd7927ff4f1191c2270a42b58f5

3- Set LP Wallet (passed):

https://testnet.bscscan.com/tx/0x3067b7ad7df18e686b1096bd4c29f5f5b8ee45ba80f4e06021ad2f9d3f5e058a

4- Set Dev Wallet (passed):

https://testnet.bscscan.com/tx/0x87542ce6b4891381f5315348925657fa4f76b4822a54aa2bf938dde19d3710b1

5- Set Transfer Fees (passed):

https://testnet.bscscan.com/tx/0x8d3f0dd678cbe20ae0cde16232b54300e2636274a82bb993bbe2f47f8a6c4489

6- Set Sell Fees (passed):

https://testnet.bscscan.com/tx/0x8369cea14622d4d14c2561d3445912531896b99da8f54605e88e76b1699cf5c7

7- Set Buy Fees (passed):

https://testnet.bscscan.com/tx/0x7b004aba4de961c0baf8a4d8e650b547b87970f86913e7abafa23fabe91fee36

MANUAL TESTING

Centralization – Buy and Sell Fees.
Severity: High
Function: setBuyFees, setSellFees
Ownership: Renounced
Status: FIXED (Ownership Renounced)

Overview:
The owner can set the buy and sell fees up to 40%, which is not recommended.

```
function setBuyFees(
    uint256 _marketingFee,
    uint256 _liquidityFee,
    uint256 _DevelopmentFee
  ) external onlyOwner {
    buyMarketingFee = _marketingFee;
    buyLPFee = _liquidityFee;
    buyDevelopmentFee = _DevelopmentFee;
    totalBuyFee = buyMarketingFee + buyLPFee + buyDevelopmentFee;
    require(totalBuyFee <= 40, "Total buy fee cannot be higher than
40%");
    emit BuyFeeUpdated(totalBuyFee, buyMarketingFee, buyLPFee,
buyDevelopmentFee);
  }
function setSellFees(
    uint256 _marketingFee,
    uint256 _liquidityFee,
    uint256 _DevelopmentFee
  ) external onlyOwner {
    sellMarketingFee = _marketingFee;
    sellLpFee = _liquidityFee;
    sellDevelopmentFee = _DevelopmentFee;
    totalSellFee = sellMarketingFee + sellLpFee + sellDevelopmentFee;
```

MANUAL TESTING

```
    require(
      totalSellFee <= 40,
      "Total sell fee cannot be higher than 40%"
    );
    emit SellFeeUpdated(totalSellFee, sellMarketingFee, sellLpFee,
sellDevelopmentFee);
```

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
Function: OpenTrading
Status: FIXED (Ownership Renounced)

Overview:
The OpenTrading 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 openTrading() external onlyOwner {
    tradingOn = true;
    swapbackEnabled = true;
    emit TradingEnabled(block.timestamp);
  }
```

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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Require Check.

Severity: High

Function: setmarketingWallet/setLPWallet/setDevWallet

Status: FIXED (Ownership Renounced)

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setMarketingWallet(address newWallet) external onlyOwner {
    emit MarketingReceiverUpdated(newWallet, MarketingReceiver);
    MarketingReceiver = newWallet;
  }
function setLPWallet(address newWallet) external onlyOwner {
    emit lpReceiverUpdated(newWallet, autoLPReceiver);
    autoLPReceiver = newWallet;
  }
function setDevWallet(address newWallet) external onlyOwner {
    emit devReceiverUpdated(newWallet, devReceiver);
    devReceiver = newWallet;
  }
```

Suggestion:

It is recommended that the address should not be able to be set as a contract address.

MANUAL TESTING

Centralization – Liquidity is added to EOA.

Severity: Medium

function: addLiquidity

Status: Open

Overview:

Liquidity is added to EOA. It may be drained by the Wallet_Liquidity.

```
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
      autoLPReceiver,
      block.timestamp
    );
  }
```

Suggestion:

It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Unsafe Usage of tx.origin

Severity: Low

Subject: Tx.origin

Status: Open

Overview:

Avoid using TX.origin for authorization, another contract can have a method that will call your contract (where the user has some funds for instance) and your contract will authorize that transaction as your address is in tx. origin.

```
 if (trasnferDelayEnabled) {
if (
        to != owner() &&
        to != address(dexRouter) &&
        to != address(dexPair)
      ) {
require(
         _holderLastTransferTimestamp[tx.origin] <
          block.number,
"_transfer:: Transfer Delay enabled. Only one purchase per block
allowed."
        );
        _holderLastTransferTimestamp[tx.origin] = block.number;
      }
    }
```

Suggestion:

You should use msg. sender for authorization (if another contract calls your contract msg.sender will be the address of the contract and not the address of the user who called the contract.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Remove Safe Math

Status: Open

Line: 10-119

Overview:

compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

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.

```
interface IDexPair {
  function sync() external;
}
```

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