auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Ted Inu

DATED : 02 APR 23'

AUDIT SUMMARY

Project name – Ted Inu

Date: 02 April, 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	1	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	1	0	0	0

USED TOOLS

Tools:

1- Manual Review:
a line by line code review has been performed by audit ace team.

2- BSC Testnet network:
all tests were done on Bsc Testnet network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: Contract has been tested on binance smart chain testnet which can be found in below link:

https://testnet.bscscan.com/token/0x230275D429cf6Aab53B0B3e584e108Ff33E8FBdB

Token Information

Token Name : Ted Inu

Token Symbol: TED

Decimals: 9

Token Supply: 300,000,000,000,000,000

Token Address:
0x308d3bAEa89B9515575aBE72a7Fd9a1667DA0149

Checksum:
b994c2483c606785413190904119a24508d351c9

Owner:
0x5521542F1CdeACB8fE0df61B18f9265924A7e1E1
(at time of audit)

TOKEN OVERVIEW

Fees:

Buy Fees: upto 30%

Sell Fees: upto 30%

Transfer Fees: upto 30%

Fees Privilige: owner

Ownership : owner

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: changing fee - changing swap threshold - excluding wallets from fees - including wallets in fees

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

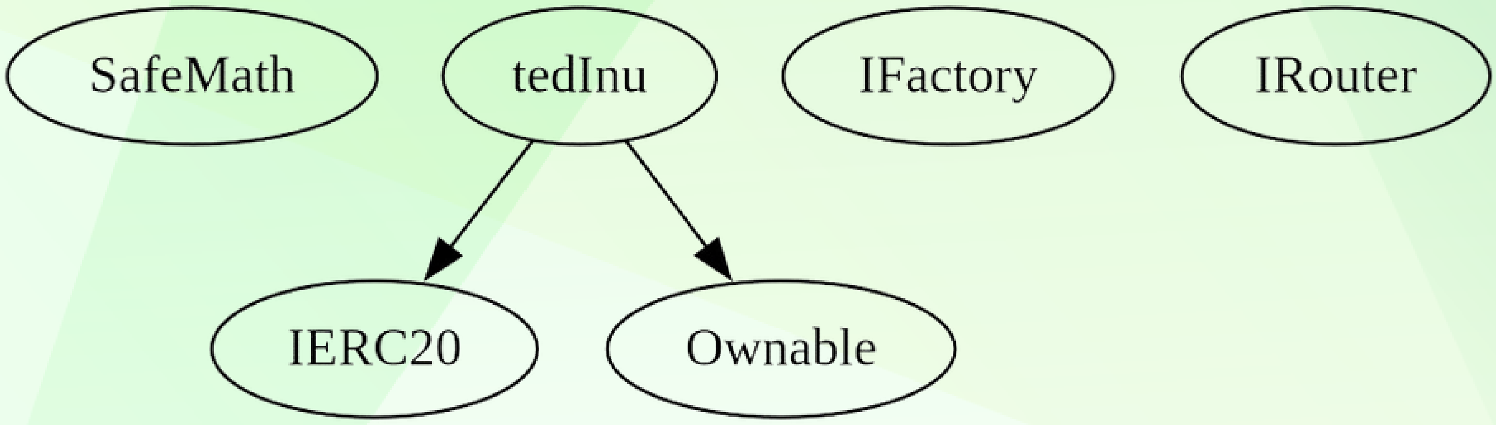
CLASSIFICATION OF RISK

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	2
🟫 High-Risk	1
🟧 Medium-Risk	0
🟨 Low-Risk	0
🟦 Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner can set buy/sell/transfer fees up to 30% each (maximum 30% fee for each transaction type).
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner cannot directly disable trades; however, it is important to note that potential vulnerabilities in the contract may be exploited by a malicious actor, which could indirectly result in the disabling of trades.
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

```
| Contract | Type | Bases | | |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **SafeMath** | Library | |||
| └ | add | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
| └ | tryAdd | Internal 🔒 | | |
| └ | trySub | Internal 🔒 | | |
| └ | tryMul | Internal 🔒 | | |
| └ | tryDiv | Internal 🔒 | | |
| └ | tryMod | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
||||||
| **IERC20** | Interface | |||
| └ | totalSupply | External ❗ | |NO❗ |
| └ | decimals | External ❗ | |NO❗ |
| └ | symbol | External ❗ | |NO❗ |
| └ | name | External ❗ | |NO❗ |
| └ | getOwner | External ❗ | |NO❗ |
| └ | balanceOf | External ❗ | |NO❗ |
| └ | transfer | External ❗ | 🛑 |NO❗ |
| └ | allowance | External ❗ | |NO❗ |
| └ | approve | External ❗ | 🛑 |NO❗ |
| └ | transferFrom | External ❗ | 🛑 |NO❗ |
||||||
| **Ownable** | Implementation | |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | isOwner | Public ❗ | |NO❗ |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
||||||
| **IFactory** | Interface | |||
| └ | createPair | External ❗ | 🛑 |NO❗ |
||||||
| **IRouter** | Interface | |||
| └ | factory | External ❗ | |NO❗ |
| └ | WETH | External ❗ | |NO❗ |
```

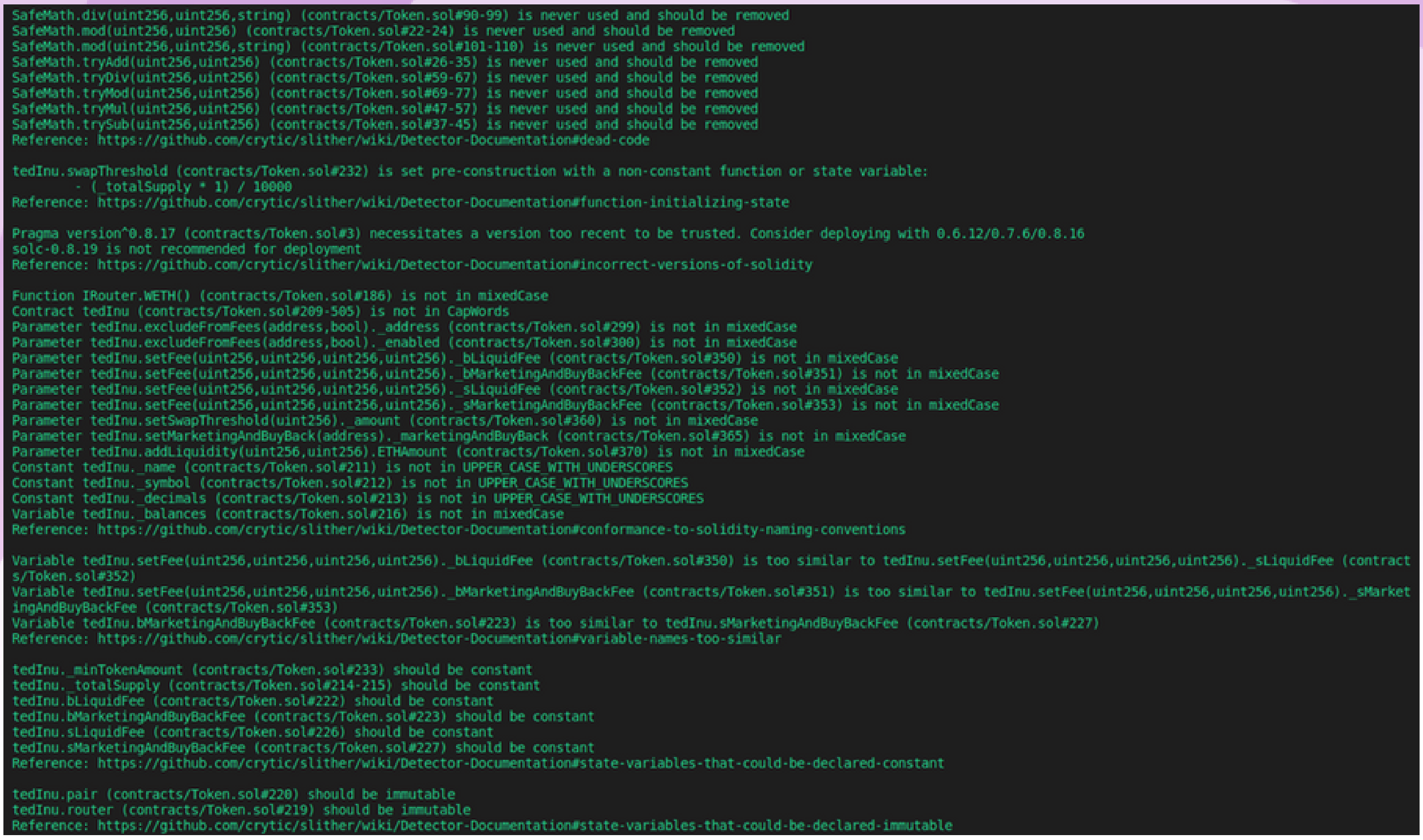
CONTRACT ASSESMENT

```
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑 |NO❗️ |
| └ | addLiquidityETH | External ❗️ | 💵 |NO❗️ |
||||||
| **tedInu** | Implementation | IERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑 | Ownable |
| └ | <Receive Ether> | External ❗️ | 💵 |NO❗️ |
| └ | name | Public ❗️ |   |NO❗️ |
| └ | symbol | Public ❗️ |   |NO❗️ |
| └ | decimals | Public ❗️ |   |NO❗️ |
| └ | getOwner | External ❗️ |   |NO❗️ |
| └ | balanceOf | Public ❗️ |   |NO❗️ |
| └ | transfer | Public ❗️ | 🛑 |NO❗️ |
| └ | allowance | Public ❗️ |   |NO❗️ |
| └ | excludeFromFees | External ❗️ | 🛑 | onlyOwner |
| └ | approve | Public ❗️ | 🛑 |NO❗️ |
| └ | totalSupply | Public ❗️ |   |NO❗️ |
| └ | checkTx | Internal 🔒 |   | |
| └ | _transfer | Private 🔐 | 🛑 | |
| └ | setFee | External ❗️ | 🛑 | onlyOwner |
| └ | setSwapThreshold | External ❗️ | 🛑 | onlyOwner |
| └ | setMarketingAndBuyBack | External ❗️ | 🛑 | onlyOwner |
| └ | addLiquidity | Private 🔐 | 🛑 | |
| └ | swapBack | Private 🔐 | 🛑 | lockTheSwap |
| └ | swapTokensForETH | Private 🔐 | 🛑 | |
| └ | swapAndLiquify | Private 🔐 | 🛑 | |
| └ | shouldSwapBack | Internal 🔒 |   | |
| └ | swapBack | Internal 🔒 | 🛑 | |
| └ | shouldTakeFee | Internal 🔒 |   | |
| └ | getTotalFee | Internal 🔒 |   | |
| └ | takeFee | Internal 🔒 | 🛑 | |
| └ | transferFrom | Public ❗️ | 🛑 |NO❗️ |
| └ | _approve | Private 🔐 | 🛑 | |
```

Legend

```
| Symbol | Meaning |
|:--------:|-----------|
```

STATIC ANALYSIS



```
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#90-99) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#22-24) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#101-110) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#26-35) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#59-67) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#69-77) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#37-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

tedInu.swapThreshold (contracts/Token.sol#232) is set pre-construction with a non-constant function or state variable:
        - (_totalSupply * 1) / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IRouter.WETH() (contracts/Token.sol#186) is not in mixedCase
Contract tedInu (contracts/Token.sol#209-505) is not in CapWords
Parameter tedInu.excludeFromFees(address,bool)._address (contracts/Token.sol#299) is not in mixedCase
Parameter tedInu.excludeFromFees(address,bool)._enabled (contracts/Token.sol#300) is not in mixedCase
Parameter tedInu.setFee(uint256,uint256,uint256,uint256)._bLiquidFee (contracts/Token.sol#350) is not in mixedCase
Parameter tedInu.setFee(uint256,uint256,uint256,uint256)._bMarketingAndBuyBackFee (contracts/Token.sol#351) is not in mixedCase
Parameter tedInu.setFee(uint256,uint256,uint256,uint256)._sLiquidFee (contracts/Token.sol#352) is not in mixedCase
Parameter tedInu.setFee(uint256,uint256,uint256,uint256)._sMarketingAndBuyBackFee (contracts/Token.sol#353) is not in mixedCase
Parameter tedInu.setSwapThreshold(uint256)._amount (contracts/Token.sol#360) is not in mixedCase
Parameter tedInu.setMarketingAndBuyBack(address)._marketingAndBuyBack (contracts/Token.sol#365) is not in mixedCase
Parameter tedInu.addLiquidity(uint256,uint256).ETHAmount (contracts/Token.sol#370) is not in mixedCase
Constant tedInu._name (contracts/Token.sol#211) is not in UPPER_CASE_WITH_UNDERSCORES
Constant tedInu._symbol (contracts/Token.sol#212) is not in UPPER_CASE_WITH_UNDERSCORES
Constant tedInu._decimals (contracts/Token.sol#213) is not in UPPER_CASE_WITH_UNDERSCORES
Variable tedInu._balances (contracts/Token.sol#216) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable tedInu.setFee(uint256,uint256,uint256,uint256)._bLiquidFee (contracts/Token.sol#350) is too similar to tedInu.setFee(uint256,uint256,uint256,uint256)._sLiquidFee (contract
s/Token.sol#352)
Variable tedInu.setFee(uint256,uint256,uint256,uint256)._bMarketingAndBuyBackFee (contracts/Token.sol#351) is too similar to tedInu.setFee(uint256,uint256,uint256,uint256)._sMarket
ingAndBuyBackFee (contracts/Token.sol#353)
Variable tedInu.bMarketingAndBuyBackFee (contracts/Token.sol#223) is too similar to tedInu.sMarketingAndBuyBackFee (contracts/Token.sol#227)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

tedInu._minTokenAmount (contracts/Token.sol#233) should be constant
tedInu._totalSupply (contracts/Token.sol#214-215) should be constant
tedInu.bLiquidFee (contracts/Token.sol#222) should be constant
tedInu.bMarketingAndBuyBackFee (contracts/Token.sol#223) should be constant
tedInu.sLiquidFee (contracts/Token.sol#226) should be constant
tedInu.sMarketingAndBuyBackFee (contracts/Token.sol#227) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

tedInu.pair (contracts/Token.sol#220) should be immutable
tedInu.router (contracts/Token.sol#219) 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 issues found

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding Liquidity (Passed):
liquidity added on Pancakeswap V2:

https://testnet.bscscan.com/tx/0xd2cd5798a159cdc32233ed6c683859a0820cdccb64a8561493366c5a94884a63

2- Buying when trading not enabled (0%)(Passed):

https://testnet.bscscan.com/tx/0x64fddb300f51a59904d615698d62780feb53fce7de9b4836117b88bbd4d58a36

3- Selling when trading not enabled (0%)(Passed):

https://testnet.bscscan.com/tx/0x67fe189425e2493a351a4121a2763c2bb3e87f27529e2f6fe8ac6c9a3ea479a2

4- Transferring when trading not enabled (0% tax) (passed):

https://testnet.bscscan.com/tx/0x79a06e999a175d5312827e9cf28ee70e902ced29f25e2ac152eabace2e268a1d

5- Buying when trading enabled (upto 30% tax) (passed):

https://testnet.bscscan.com/tx/0x244168b556650e0a60f25845514db253c0c39d080f57bc3bd7ecf6ed25a2823e

FUNCTIONAL TESTING

6- Selling when trading enabled (upto 30% tax) (passed):

https://testnet.bscscan.com/tx/0xf2b7036a2d64e331937d037a8cdeed8acbd871e970b6de2e7f2962cc18a18df8

7- Transferring when trading enabled (upto 30% tax) (passed):

https://testnet.bscscan.com/tx/0x7b131a003ad6d0be11cd5cbbda2ec7b082fe5c5fdd3e8103fcda43489c1b5f39

8- Internal swap (passed):
As can seen in this transaction, marketing wallet received TED Tokens
https://testnet.bscscan.com/token/0x230275d429cf6aab53b0b3e584e108ff33e8fbdb?a=0x6820f85a61eaef5c43c21cc52da5295a3a9b735e

9- Auto Liquidity (passed):
Auto liquidity generated tokens are burnt
https://testnet.bscscan.com/token/0xf9a584a018b321c784c60c50ccb36127291f1b4f?a=0x000000000000000000000000000000000000dead

MANUAL TESTING

Logical - Sell tax disables trades when sellFee is set to 0

Severity: Critical
Function: setFee - swapBack
Lines: 386
Status: No Resolved
If the sellFee is set to 0, the sell transaction will fail at the swapBack function, as a division by zero occurs. Additionally.

```
function swapBack(uint256 tokens) private lockTheSwap {
   uint256 tokensForLiquid = tokens.mul(sLiquidFee).div(sellFee);
   uint256 tokensForMarketingandBuyBack = tokens.sub(tokensForLiquid);
   swapAndLiquify(tokensForLiquid);
   _transfer(
      address(this),
      marketingAndBuyBack,
      tokensForMarketingandBuyBack
   );
}
```

Recommendation:
Implement safety checks to make sure a zero division does not happen, this can be done by returning from swapBack function if **sellFee** is zero.

MANUAL TESTING

Logical - Setting internal swap threshold to 0 can disable sells

Severity: Critical
Function: setSwapThreshold
Lines: 360
Status: No Resolved
If the swapThreshold is set to 0, sell transactions will fail at the swapBack function. This occurs because the checks for performing a swapAndLiquify will still pass even if the swapThreshold is set to 0 and the contract has 0 tokens. Consequently, the transaction will fail while attempting to swap 0 tokens (i.e., swapThreshold) to BNB. Additionally, setting the swapThreshold to an excessively large number leads to a high slippage percentage during sell transactions.

```
function swapBack(
   address sender,
   address recipient,
   uint256 amount
) internal {
   if (shouldSwapBack(sender, recipient, amount)) {
      swapBack(swapThreshold);
   }
}

function swapBack(uint256 tokens) private lockTheSwap {
   uint256 tokensForLiquid = tokens.mul(sLiquidFee).div(sellFee);
   uint256 tokensForMarketingandBuyBack = tokens.sub(tokensForLiquid);
   swapAndLiquify(tokensForLiquid);
   _transfer(
      address(this),
      marketingAndBuyBack,
      tokensForMarketingandBuyBack
   );
}
```

Recommendation:
Ensure that the swapThreshold is set to a value greater than a reasonable minimum and less than a reasonable maximum. This will help prevent issues related to disabled sell transactions or high slippage percentages during trades.

MANUAL TESTING

Centralization – Excessive max buy/sell/transfer fees

Severity: **High**
Function: setFee
Lines: 349
Status: No Resolved
The owner has the ability to set up to 30% tax for buys and 30% tax for sells (and transfers), which can result in a 60% total tax in a buy and then sell transaction if both fees are set to their maximum value. These high fees can negatively impact the token's economy and make trading the token unprofitable for investors.

```
function setFee(
   uint256 _bLiquidFee,
   uint256 _bMarketingAndBuyBackFee,
   uint256 _sLiquidFee,
   uint256 _sMarketingAndBuyBackFee
) external onlyOwner {
   buyFee = _bLiquidFee.add(_bMarketingAndBuyBackFee);
   sellFee = _sLiquidFee.add(_sMarketingAndBuyBackFee);
   require(buyFee <= 30 && sellFee <= 30, "Must keep fees at 30% or less");
}
```

Recommendation:
In accordance with Pinksale's safu criteria, it is recommended to set a more reasonable tax limit, such as a maximum of 24% for the combined buy and sell fees. This would help maintain a healthier token economy and encourage more investors to trade the token.

DISCLAIMER

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