auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

DADA

DATED : 24 Jan, 2024

MANUAL TESTING

Centralization – Enabling Trades

Severity: High
Function: startTrade
Status: Open
Overview:
The **startTrade** 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 startTrade() external onlyOwner {
        require(0 == startTradeBlock, "trading");
        startTradeBlock = block.number;
    }
```

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 – The owner can lock the token

Severity: High

Function: setLimitAmount, setTxLimitAmount, setHolderRewardCondition and SetHolderCondition.

Status: Open

Overview:

In this setLimitAmount, setTxLimitAmount, setHolderRewardCondition and SetHolderCondition.

```
function setLimitAmount(uint256 amount) external onlyOwner {
        _limitAmount = amount * 10 ** _decimals;
    }
function setTxLimitAmount(uint256 amount) external onlyOwner {
        _txLimitAmount = amount * 10 ** _decimals;
    }
 function setMinTotal(uint256 total) external onlyOwner {
        _minTotal = total * 10 ** _decimals;
    }
function setHolderRewardCondition(uint256 amount) external onlyOwner {
        holderRewardCondition = amount;
    }
function setHolderCondition(uint256 amount) external onlyOwner {
        holderCondition = amount;
    }
```

Suggestion:

It is recommended that there be a required check for zero address.

MANUAL TESTING

Centralization – Buy, Sell and Transfer Fees

Severity: High

Function: setBuyFee, setSellFee and setTransferFee

Status: Open

Overview:

The owner can set the buy and sell fees up to 100%, which is not recommended.

```
function setBuyFee(
        uint256 buyDestroyFee, uint256 buyFundFee, uint256 buyRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _buyDestroyFee = buyDestroyFee;
        _buyFundFee = buyFundFee;
        _buyRewardFee = buyRewardFee;
        _buyLPDividendFee = lpDividendFee;
        _buyLPFee = lpFee;
    }
function setSellFee(
        uint256 sellDestroyFee, uint256 sellFundFee, uint256 sellRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _sellDestroyFee = sellDestroyFee;
        _sellFundFee = sellFundFee;
        _sellRewardFee = sellRewardFee;
        _sellLPDividendFee = lpDividendFee;
        _sellLPFee = lpFee;
    }
function setTransferFee(uint256 fee) external onlyOwner {
        _transferFee = fee;
    }
```

Suggestion:

It is recommended that no fees in the contract should be more than 25% of the contract.

Centralization – Lp, Add and Remove LPFees.

Severity: High
Function: setLPFeeReceiver, setAddLPFee and setRemoveLPFee
Status: Open
Overview:
The owner can set the buy and sell fees up to 100%, which is not recommended.

```
function setLPFeeReceiver(address adr) external onlyOwner {
        _lpFeeReceiver = adr;
        _feeWhiteList[adr] = true;
    }
function setAddLPFee(uint256 fee) external onlyOwner {
        _addLPFee = fee;
    }
function setRemoveLPFee(uint256 fee) external onlyOwner {
        _removeLPFee = fee;
    }
```

Suggestion:
It is recommended that no fees in the contract should be more than 25% of the contract.

AUDIT SUMMARY

Project name – DADA

Date: 24 Jan, 2024

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

Audit Status: **High Risk Major Flag**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	4	0	3	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/0xf4c423970847161454078c545f97c0c3be7f33a8#code

Token Information

Token Name : DADA

Token Symbol: DADA

Decimals: 9

Token Supply: 420690000000000000

Network: Binance Smart Chain

Token Type: BEP-20

Token Address:
0x490bE8605051c4876e4A94910a941e3549801D74

Checksum:
AEde641126e217b2b455d49e77fc41223

Owner:
0x08b52556eF45eD6E5F43e7e1D61A9C62592E11ed
(at time of writing the audit)

Deployer:
0x08b52556eF45eD6E5F43e7e1D61A9C62592E11ed

TOKEN OVERVIEW

Fees:

Buy Fee: 40-100% (High)

Sell Fee: 40-100% (High)

Transfer Fee: 100% (High)

Fees Privilege: Owner

Ownership: Owned

Minting: No

Max Tx Amount/ Max Wallet Amount: 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

STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
AbsToken.swapTokenForFund(uint256) (DADA.sol#504-551) uses arbitrary from in transferFrom: USDT.transferFrom(tokenDistributor,address(this),usdtBalance) (DA
DA.sol#533)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
AbsToken.swapTokenForFund(uint256) (DADA.sol#504-551) ignores return value by USDT.transferFrom(tokenDistributor,address(this),usdtBalance) (DADA.sol#533)
AbsToken.swapTokenForFund(uint256) (DADA.sol#504-551) ignores return value by USDT.transfer(fundAddress,fundUsdt) (DADA.sol#537)
AbsToken.swapTokenForFund(uint256) (DADA.sol#504-551) ignores return value by USDT.transfer(rewardAddress,fundUsdt2) (DADA.sol#542)
AbsToken.claimToken(address,uint256) (DADA.sol#670-674) ignores return value by IERC20(token).transfer(fundAddress,amount) (DADA.sol#672)
AbsToken.processReward(uint256) (DADA.sol#718-769) ignores return value by usdt.transfer(shareHolder,amount) (DADA.sol#758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (DADA.sol#410-502) performs a multiplication on the result of a division:
        - feeAmount = tAmount * _transferFee / 10000 (DADA.sol#486)
        - swapAmount = 2 * feeAmount (DADA.sol#490)
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (DADA.sol#410-502) performs a multiplication on the result of a division:
        - fundAmount_scope_4 = tAmount * (_sellFundFee + _sellRewardFee + _sellLPDividendFee + _sellLPFee) / 10000 (DADA.sol#469)
        - numTokensSellToFund = fundAmount_scope_4 * 230 / 100 (DADA.sol#477)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (DADA.sol#410-502):
        External calls:
        - _tokenTransfer(tokenDistributor,address(this),swapAmount,false,false,false) (DADA.sol#495)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,fundAddress,block.timestamp) (DADA.sol#561-567)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,tokenDistributor,block.timestamp) (DADA.so
l#523-529)
                - USDT.transferFrom(tokenDistributor,address(this),usdtBalance) (DADA.sol#533)
                - USDT.transfer(fundAddress,fundUsdt) (DADA.sol#537)
                - USDT.transfer(rewardAddress,fundUsdt2) (DADA.sol#542)
                - _swapRouter.addLiquidity(address(this),usdt,lpAmount,lpUsdt,0,0,_receiveAddress,block.timestamp) (DADA.sol#547-549)
        - swapTokenForFund2(swapAmount) (DADA.sol#496)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,fundAddress,block.timestamp) (DADA.sol#561-567)
        State variables written after the call(s):
        - swapTokenForFund2(swapAmount) (DADA.sol#496)
                - inSwap = true (DADA.sol#168)
                - inSwap = false (DADA.sol#170)
        AbsToken.inSwap (DADA.sol#126) can be used in cross function reentrancies:
        - AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (DADA.sol#410-502)
        - AbsToken.lockTheSwap() (DADA.sol#167-171)
```

```
INFO:Detectors:
AbsToken._transfer(address,address,uint256).takeFee (DADA.sol#287) is a local variable never initialized
AbsToken._transfer(address,address,uint256).isAddLP (DADA.sol#311) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).maxDestroyAmount_scope_3 (DADA.sol#457) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).feeAmount (DADA.sol#419) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).maxDestroyAmount (DADA.sol#433) is a local variable never initialized
AbsToken._transfer(address,address,uint256).isRemoveLP (DADA.sol#312) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
TokenDistributor.constructor(address) (DADA.sol#100-102) ignores return value by IERC20(token).approve(msg.sender,uint256(~ uint256(0))) (DADA.sol#101)
AbsToken.constructor(address,address,string,string,uint8,uint256,address,address,address,uint256,uint256,uint256) (DADA.sol#173-229) ignores return value by
 IERC20(usdt).approve(address(swapRouter),MAX) (DADA.sol#185)
AbsToken._isAddLiquidity(uint256) (DADA.sol#357-378) ignores return value by (r0,r1) = mainPair.getReserves() (DADA.sol#359)
AbsToken._isRemoveLiquidity() (DADA.sol#380-394) ignores return value by (r0,r1) = mainPair.getReserves() (DADA.sol#382)
AbsToken.swapTokenForFund(uint256) (DADA.sol#504-551) ignores return value by _swapRouter.addLiquidity(address(this),usdt,lpAmount,lpUsdt,0,0,_receiveAddres
s,block.timestamp) (DADA.sol#547-549)
AbsToken.getTokenPrice() (DADA.sol#808-824) ignores return value by (reserve0,reserve1) = swapPair.getReserves() (DADA.sol#810)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
AbsToken.allowance(address,address).owner (DADA.sol#257) shadows:
        - Ownable.owner() (DADA.sol#78-80) (function)
AbsToken._approve(address,address,uint256).owner (DADA.sol#274) shadows:
        - Ownable.owner() (DADA.sol#78-80) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
AbsToken.setFundAddress(address).addr (DADA.sol#579) lacks a zero-check on :
                - fundAddress = addr (DADA.sol#580)
AbsToken.setRewardAddress(address).addr (DADA.sol#584) lacks a zero-check on :
                - rewardAddress = addr (DADA.sol#585)
AbsToken.setReceiveAddress(address).addr (DADA.sol#589) lacks a zero-check on :
                - _receiveAddress = addr (DADA.sol#590)
AbsToken.setLPFeeReceiver(address).adr (DADA.sol#795) lacks a zero-check on :
                - _lpFeeReceiver = adr (DADA.sol#796)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
AbsToken.addHolder(address) (DADA.sol#698-710) uses assembly
        - INLINE ASM (DADA.sol#702)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AbsToken._transfer(address,address,uint256) (DADA.sol#279-355) has a high cyclomatic complexity (17).
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (DADA.sol#410-502) has a high cyclomatic complexity (22).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Variable AbsToken._rewardList (DADA.sol#117) is not in mixedCase
Variable AbsToken._DADA (DADA.sol#118) is not in mixedCase
Variable AbsToken._buyDestroyFee (DADA.sol#131) is not in mixedCase
Variable AbsToken._buyFundFee (DADA.sol#132) is not in mixedCase
Variable AbsToken._buyRewardFee (DADA.sol#133) is not in mixedCase
Variable AbsToken._buyLPDividendFee (DADA.sol#134) is not in mixedCase
Variable AbsToken._buyLPFee (DADA.sol#135) is not in mixedCase
Variable AbsToken._sellDestroyFee (DADA.sol#137) is not in mixedCase
Variable AbsToken._sellFundFee (DADA.sol#138) is not in mixedCase
Variable AbsToken._sellRewardFee (DADA.sol#139) is not in mixedCase
Variable AbsToken._sellLPDividendFee (DADA.sol#140) is not in mixedCase
Variable AbsToken._sellLPFee (DADA.sol#141) is not in mixedCase
Variable AbsToken._transferFee (DADA.sol#143) is not in mixedCase
Variable AbsToken._mainPair (DADA.sol#149) is not in mixedCase
Variable AbsToken._limitAmount (DADA.sol#151) is not in mixedCase
Variable AbsToken._txLimitAmount (DADA.sol#152) is not in mixedCase
Variable AbsToken._minTotal (DADA.sol#153) is not in mixedCase
Variable AbsToken._receiveAddress (DADA.sol#155) is not in mixedCase
Variable AbsToken._airdropLen (DADA.sol#158) is not in mixedCase
Variable AbsToken._airdropAmount (DADA.sol#159) is not in mixedCase
Variable AbsToken._removeLPFee (DADA.sol#161) is not in mixedCase
Variable AbsToken._addLPFee (DADA.sol#162) is not in mixedCase
Variable AbsToken._lpFeeReceiver (DADA.sol#163) is not in mixedCase
Constant AbsToken._killBlock (DADA.sol#165) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Variable ISwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (DADA.sol#46) is too similar to ISwapRoute
r.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (DADA.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
AbsToken._transfer(address,address,uint256) (DADA.sol#279-355) uses literals with too many digits:
        - maxSellAmount = balance * 99999 / 100000 (DADA.sol#290)
AbsToken._transfer(address,address,uint256) (DADA.sol#279-355) uses literals with too many digits:
        - processReward(500000) (DADA.sol#352)
DADA.constructor() (DADA.sol#828-843) uses literals with too many digits:
        - AbsToken(address(0x10ED43C718714eb63d5aA57B78B54704E256024E),address(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c),DADA,DADA,9,420690000000000000,ad
dress(0x8dD1d29524BD2c11deC7Bb144393cAb2d15Bb9b0),address(0x8dD1d29524BD2c11deC7Bb144393cAb2d15Bb9b0),address(0x08b52556eF45eD6E5F43e7e1D61A9C62592E11ed),42
0690000000000000,0,420690000000000000) (DADA.sol#828-841)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
AbsToken._mainPair (DADA.sol#149) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:DADA.sol analyzed (8 contracts with 93 detectors), 64 result(s) found
```

INHERITANCE TREE

Ownership Privileges

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set Fund/Reward/Receive addresses.
- The owner can set the Buy/Sell/Transfer fee to more than 100%.
- The owner can start trading.
- The owner whitelist addresses.
- The owner can start DADA.
- The owner can set the Reward list address.
- The owner can set a swap pair list address.
- The owner can set a limit amount.
- The owner can set the Tx limit amount.
- The owner can set the min total.
- The owner can set reward conditions.
- The owner can set holder conditions.
- The owner can exclude the holder's address.
- The owner can set the LPReceiver/AddLP/RemoveLP fee to more than 100%.

Functional Tests

1- Approve (passed):

https://testnet.bscscan.com/tx/0xba78524e5271d24cf93b645918f869e6b643b751f3d4932c15dfadd159dcd256

2- Set Buy Fee (passed):

https://testnet.bscscan.com/tx/0xc16f95121fcb934169b7599dd7cfad2aa119ecd1a05d0493c84a5ed28b171073

3- Set Sell Fee (passed):

https://testnet.bscscan.com/tx/0xbef4352379175beccf3138b3dbeb3e73b12f5cd85f518d487dd5a629b00d1800

4- Start Trade (passed):

https://testnet.bscscan.com/tx/0x6ab3b406a1ac6bdb2fe886009d01a12366d3b4adda102d2fb403bd31942dd04f

5- Set Transfer Fee (passed):

https://testnet.bscscan.com/tx/0xc4d24264d970ed1992fe9c0818b26b0b47068504bc2255eabf5217d2cbd757c3

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	4
Medium-Risk	0
Low-Risk	3
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Enabling Trades

Severity: High
Function: startTrade
Status: Open
Overview:
The **startTrade** 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 startTrade() external onlyOwner {
        require(0 == startTradeBlock, "trading");
        startTradeBlock = block.number;
    }
```

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 – The owner can lock the token

Severity: High

Function: setLimitAmount, setTxLimitAmount, setHolderRewardCondition and SetHolderCondition.

Status: Open

Overview:

In this setLimitAmount, setTxLimitAmount, setHolderRewardCondition and SetHolderCondition.

```
function setLimitAmount(uint256 amount) external onlyOwner {
        _limitAmount = amount * 10 ** _decimals;
    }
function setTxLimitAmount(uint256 amount) external onlyOwner {
        _txLimitAmount = amount * 10 ** _decimals;
    }
 function setMinTotal(uint256 total) external onlyOwner {
        _minTotal = total * 10 ** _decimals;
    }
function setHolderRewardCondition(uint256 amount) external onlyOwner {
        holderRewardCondition = amount;
    }
function setHolderCondition(uint256 amount) external onlyOwner {
        holderCondition = amount;
    }
```

Suggestion:

It is recommended that there be a required check for zero address.

MANUAL TESTING

Centralization – Buy, Sell and Transfer Fees

Severity: High

Function: setBuyFee, setSellFee and setTransferFee

Status: Open

Overview:

The owner can set the buy and sell fees up to 100%, which is not recommended.

```
function setBuyFee(
        uint256 buyDestroyFee, uint256 buyFundFee, uint256 buyRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _buyDestroyFee = buyDestroyFee;
        _buyFundFee = buyFundFee;
        _buyRewardFee = buyRewardFee;
        _buyLPDividendFee = lpDividendFee;
        _buyLPFee = lpFee;
    }
function setSellFee(
        uint256 sellDestroyFee, uint256 sellFundFee, uint256 sellRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _sellDestroyFee = sellDestroyFee;
        _sellFundFee = sellFundFee;
        _sellRewardFee = sellRewardFee;
        _sellLPDividendFee = lpDividendFee;
        _sellLPFee = lpFee;
    }
function setTransferFee(uint256 fee) external onlyOwner {
        _transferFee = fee;
    }
```

Suggestion:

It is recommended that no fees in the contract should be more than 25% of the contract.

MANUAL TESTING

Centralization – Lp, Add and Remove LPFees.

Severity: High
Function: setLPFeeReceiver, setAddLPFee and setRemoveLPFee
Status: Open
Overview:
The owner can set the buy and sell fees up to 100%, which is not recommended.

```
function setLPFeeReceiver(address adr) external onlyOwner {
        _lpFeeReceiver = adr;
        _feeWhiteList[adr] = true;
    }
function setAddLPFee(uint256 fee) external onlyOwner {
        _addLPFee = fee;
    }
function setRemoveLPFee(uint256 fee) external onlyOwner {
        _removeLPFee = fee;
    }
```

Suggestion:
It is recommended that no fees in the contract should be more than 25% of the contract.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Function: 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 setFundAddress(address addr) external onlyOwner {
        fundAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setRewardAddress(address addr) external onlyOwner {
        rewardAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setReceiveAddress(address addr) external onlyOwner {
        _receiveAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setBuyFee(
        uint256 buyDestroyFee, uint256 buyFundFee, uint256 buyRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _buyDestroyFee = buyDestroyFee;
```

MANUAL TESTING

```
        _buyFundFee = buyFundFee;
        _buyRewardFee = buyRewardFee;
        _buyLPDividendFee = lpDividendFee;
        _buyLPFee = lpFee;
    }
function setSellFee(
        uint256 sellDestroyFee, uint256 sellFundFee, uint256 sellRewardFee,
        uint256 lpDividendFee, uint256 lpFee
    ) external onlyOwner {
        _sellDestroyFee = sellDestroyFee;
        _sellFundFee = sellFundFee;
        _sellRewardFee = sellRewardFee;
        _sellLPDividendFee = lpDividendFee;
        _sellLPFee = lpFee;
    }
function setTransferFee(uint256 fee) external onlyOwner {
        _transferFee = fee;
    }
function startDADA() external onlyOwner {
        require(0 == startDADABlock, "startDADA");
        startDADABlock = block.number;
    }
function startTrade() external onlyOwner {
        require(0 == startTradeBlock, "trading");
        startTradeBlock = block.number;
    }
function setDADA(address addr, bool enable) external onlyOwner {
        _DADA[addr] = enable;
    }
function batchSetDADA(address [] memory addr, bool enable) external onlyOwner {
        for (uint i = 0; i < addr.length; i++) {
            _DADA[addr[i]] = enable;
        }
    }
function setSwapPairList(address addr, bool enable) external onlyOwner {
        _swapPairList[addr] = enable;
    }
```

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Status: Open

Overview:

functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function setFundAddress(address addr) external onlyOwner {
        fundAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setRewardAddress(address addr) external onlyOwner {
        rewardAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setReceiveAddress(address addr) external onlyOwner {
        _receiveAddress = addr;
        _feeWhiteList[addr] = true;
    }
function setDADA(address addr, bool enable) external onlyOwner {
        _DADA[addr] = enable;
    }
function setSwapPairList(address addr, bool enable) external onlyOwner {
        _swapPairList[addr] = enable;
    }
```

Suggestion:

It is suggested that the address should not be zero or dead.

MANUAL TESTING

Centralization – Local Variable Shadowing

Severity: Low

Function: _approve and allowance

Status: Open

Overview:

```
function allowance(address owner, address spender) public view override returns
(uint256) {
        return _allowances[owner][spender];
    }
function _approve(address owner, address spender, uint256 amount) private {
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
```

Suggestion:

Rename the local variable that shadows another component

MANUAL TESTING

Optimization

Severity: Informational
Subject: Old Pragma Solidity version
Status: Open
Overview:
It is considered best practice to pick the latest compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.18;
```

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