auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

ChuanJG

DATED : 11 June 24'

MANUAL TESTING

Centralization – Transfer Fees.

Severity: High

Function: setTransferFee

Status: Open

Overview:

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

```
function setTransferFee(uint256 fee) external onlyOwner {
    require(_transferFee < 2050, "fee too high");
   _transferFee = fee;
 }
```

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: startTrading

Status: Open

Overview:

The startTrading 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 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: Open

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 setFundAddress(address addr) external onlyOwner {
    fundAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress2(address addr) external onlyOwner {
    fundAddress2 = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress3(address addr) external onlyOwner {
    fundAddress3 = addr;
    _feeWhiteList[addr] = true;
  }
function setReceiveAddress(address addr) external onlyOwner {
    _receiveAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setLPFeeReceiver(address adr) external onlyOwner {
    _lpFeeReceiver = adr;
    _feeWhiteList[adr] = true;
  }
```

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

AUDIT SUMMARY

Project name – ChuanJG

Date: 11 June, 2024

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

Audit Status: **HIGH RISK MAJOR FLAGS**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	3	1	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/0xfd4c8a47d2e77a44134da821f161ee154c5c4a28#code

Token Information

Token Address:
0x652dA58A120dC323E9f5ff06Df9b88563f0a7C26

Name: ChuanJG

Symbol: ChuanJG

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0xC1Dc381575b0eaeD97aD535AEfa1F0a8DfaDEc21

Deployer: 0xC1Dc381575b0eaeD97aD535AEfa1F0a8DfaDEc21

Token Supply: 10000000000

Checksum: A17acbefe2a12642d388659dffd20231

Testnet:
https://testnet.bscscan.com/address/0xfd4c8a47d2e77a44134da821f161ee154c5c4a28#code

TOKEN OVERVIEW

Buy Fee: 21.5%

Sell Fee: 21.5%

Transfer Fee: 25%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

AUDIT METHODOLOGY

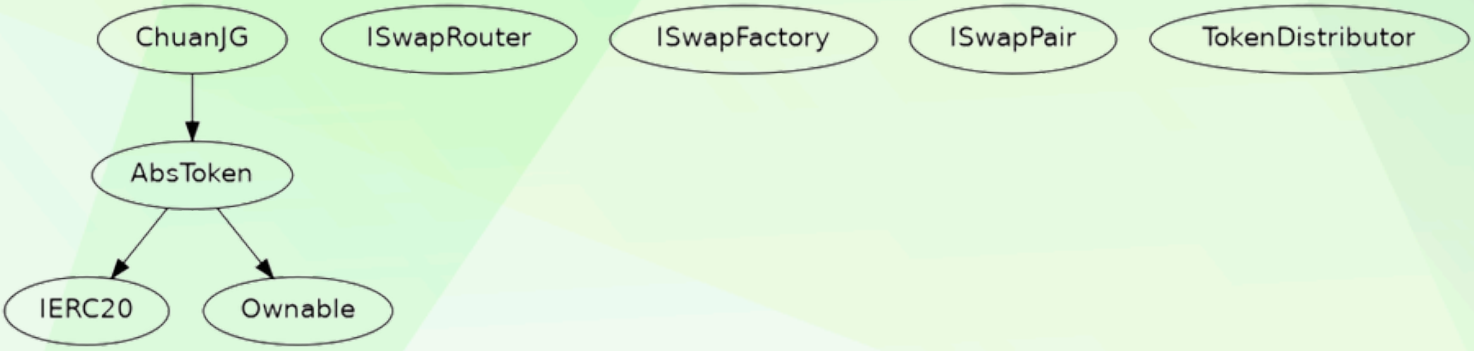
The auditing process will follow a routine as special considerations by Auditace:

- 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.

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

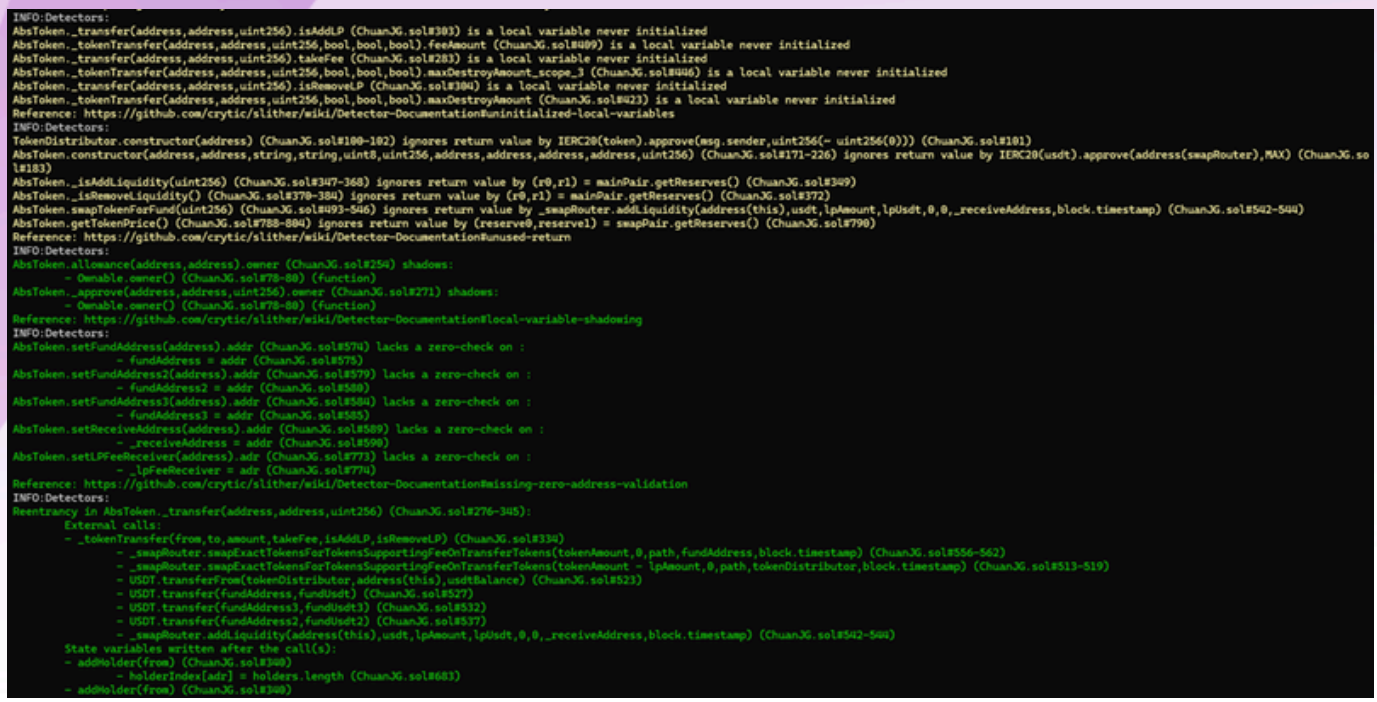
INHERITANCE TREE



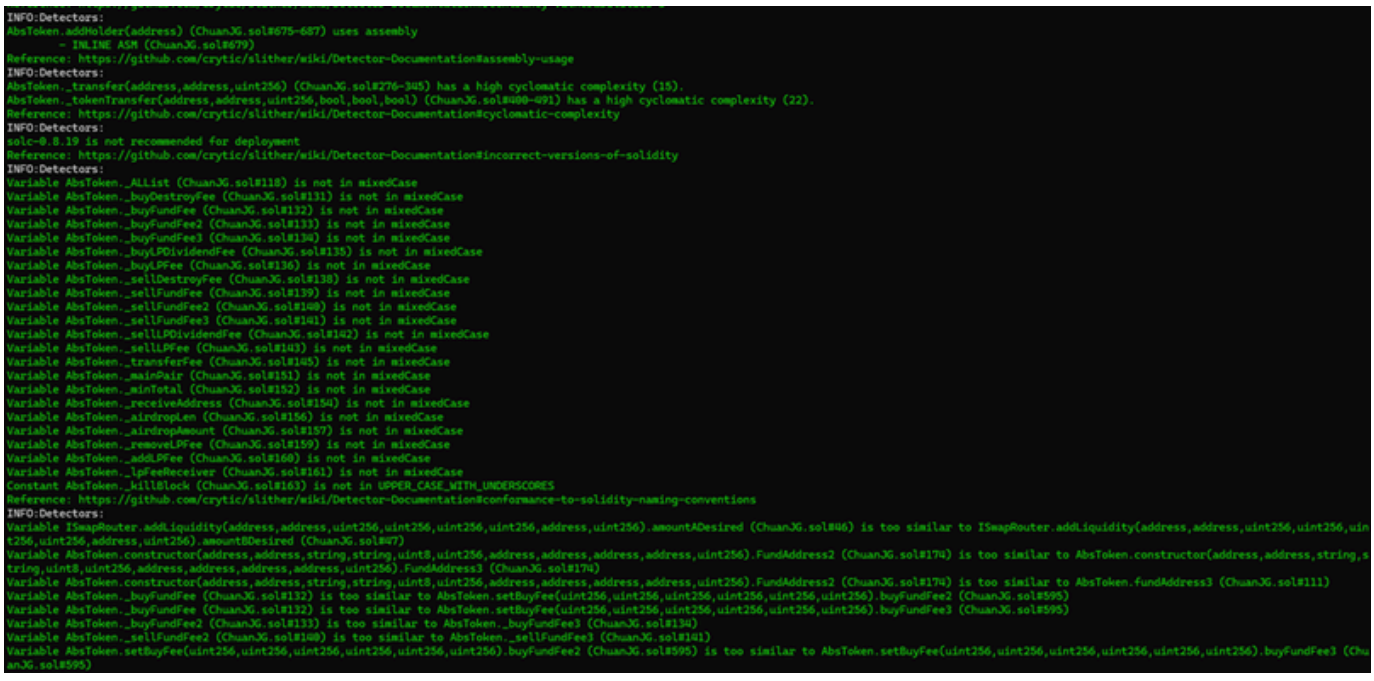
POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can start trading.
- The owner can set a buy/sell fee not to more than 21%.
- The owner can set a transfer fee/ setAddLPFee of more than 25%.
- The owner can set a swap pair list.
- The owner can claim a token.
- The owner can set holder reward conditions.
- The owner can exclude address.
- The owner set up an Automated market maker pair.
- The owner can set setFundAddress/ setFundAddress2/ setFundAddress3/setReceiveAddress/setLPReceiver.

STATIC ANALYSIS

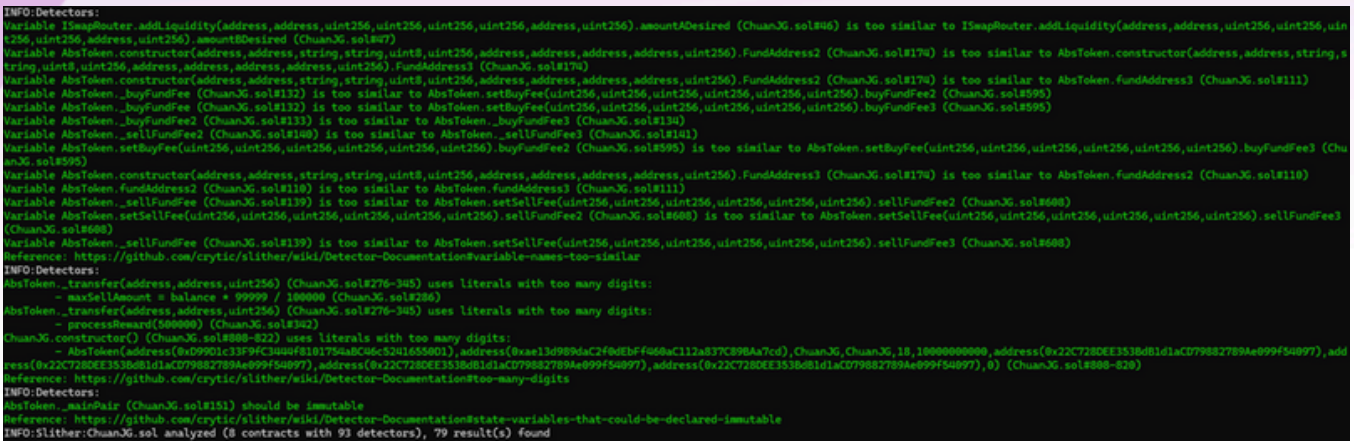


```
INFO:Detectors:
AbsToken._transfer(address,address,uint256).isAddLP (ChuanJG.sol#303) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).feeAmount (ChuanJG.sol#489) is a local variable never initialized
AbsToken._transfer(address,address,uint256).takeFee (ChuanJG.sol#283) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).maxDestroyAmount_scope_3 (ChuanJG.sol#446) is a local variable never initialized
AbsToken._transfer(address,address,uint256).isRemoveLP (ChuanJG.sol#304) is a local variable never initialized
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool).maxDestroyAmount (ChuanJG.sol#423) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
TokenDistributor.constructor(address) (ChuanJG.sol#100-102) ignores return value by IERC20(token).approve(msg.sender,uint256(~ uint256(0))) (ChuanJG.sol#101)
AbsToken.constructor(address,address,string,string,uint8,uint256,address,address,address,address,uint256) (ChuanJG.sol#171-226) ignores return value by IERC20(usdt).approve(address(swapRouter),MAX) (ChuanJG.so
l#183)
AbsToken._isAddLiquidity(uint256) (ChuanJG.sol#347-368) ignores return value by (r0,r1) = mainPair.getReserves() (ChuanJG.sol#349)
AbsToken._isRemoveLiquidity() (ChuanJG.sol#370-384) ignores return value by (r0,r1) = mainPair.getReserves() (ChuanJG.sol#372)
AbsToken.swapTokenForFund(uint256) (ChuanJG.sol#493-546) ignores return value by _swapRouter.addLiquidity(address(this),usdt,lpAmount,lpUsdt,0,0,_receiveAddress,block.timestamp) (ChuanJG.sol#542-544)
AbsToken.getTokenPrice() (ChuanJG.sol#788-804) ignores return value by (reserve0,reserve1) = swapPair.getReserves() (ChuanJG.sol#790)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
AbsToken.allowance(address,address).owner (ChuanJG.sol#254) shadows:
        - Ownable.owner() (ChuanJG.sol#78-80) (function)
AbsToken._approve(address,address,uint256).owner (ChuanJG.sol#271) shadows:
        - Ownable.owner() (ChuanJG.sol#78-80) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
AbsToken.setFundAddress(address).addr (ChuanJG.sol#574) lacks a zero-check on :
        - fundAddress = addr (ChuanJG.sol#575)
AbsToken.setFundAddress2(address).addr (ChuanJG.sol#579) lacks a zero-check on :
        - fundAddress2 = addr (ChuanJG.sol#580)
AbsToken.setFundAddress3(address).addr (ChuanJG.sol#584) lacks a zero-check on :
        - fundAddress3 = addr (ChuanJG.sol#585)
AbsToken.setReceiveAddress(address).addr (ChuanJG.sol#589) lacks a zero-check on :
        - _receiveAddress = addr (ChuanJG.sol#590)
AbsToken.setLPFeeReceiver(address).adr (ChuanJG.sol#773) lacks a zero-check on :
        - _lpFeeReceiver = adr (ChuanJG.sol#774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in AbsToken._transfer(address,address,uint256) (ChuanJG.sol#276-345):
        External calls:
        - _tokenTransfer(from,to,amount,takeFee,isAddLP,isRemoveLP) (ChuanJG.sol#334)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,fundAddress,block.timestamp) (ChuanJG.sol#556-562)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,tokenDistributor,block.timestamp) (ChuanJG.sol#513-519)
                - USDT.transferFrom(tokenDistributor,address(this),usdtBalance) (ChuanJG.sol#523)
                - USDT.transfer(fundAddress,fundUsdt) (ChuanJG.sol#527)
                - USDT.transfer(fundAddress3,fundUsdt3) (ChuanJG.sol#532)
                - USDT.transfer(fundAddress2,fundUsdt2) (ChuanJG.sol#537)
                - _swapRouter.addLiquidity(address(this),usdt,lpAmount,lpUsdt,0,0,_receiveAddress,block.timestamp) (ChuanJG.sol#542-544)
        State variables written after the call(s):
        - addHolder(from) (ChuanJG.sol#340)
                - holderIndex[adr] = holders.length (ChuanJG.sol#683)
        - addHolder(from) (ChuanJG.sol#340)
```



```
INFO:Detectors:
AbsToken.addHolder(address) (ChuanJG.sol#675-687) uses assembly
        - INLINE ASM (ChuanJG.sol#679)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AbsToken._transfer(address,address,uint256) (ChuanJG.sol#276-345) has a high cyclomatic complexity (15).
AbsToken._tokenTransfer(address,address,uint256,bool,bool,bool) (ChuanJG.sol#480-491) has a high cyclomatic complexity (22).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable AbsToken._ALList (ChuanJG.sol#118) is not in mixedCase
Variable AbsToken._buyDestroyFee (ChuanJG.sol#131) is not in mixedCase
Variable AbsToken._buyFundFee (ChuanJG.sol#132) is not in mixedCase
Variable AbsToken._buyFundFee2 (ChuanJG.sol#133) is not in mixedCase
Variable AbsToken._buyFundFee3 (ChuanJG.sol#134) is not in mixedCase
Variable AbsToken._buyLPDividendFee (ChuanJG.sol#135) is not in mixedCase
Variable AbsToken._buyLPFee (ChuanJG.sol#136) is not in mixedCase
Variable AbsToken._sellDestroyFee (ChuanJG.sol#138) is not in mixedCase
Variable AbsToken._sellFundFee (ChuanJG.sol#139) is not in mixedCase
Variable AbsToken._sellFundFee2 (ChuanJG.sol#140) is not in mixedCase
Variable AbsToken._sellFundFee3 (ChuanJG.sol#141) is not in mixedCase
Variable AbsToken._sellLPDividendFee (ChuanJG.sol#142) is not in mixedCase
Variable AbsToken._sellLPFee (ChuanJG.sol#143) is not in mixedCase
Variable AbsToken._transferFee (ChuanJG.sol#145) is not in mixedCase
Variable AbsToken._mainPair (ChuanJG.sol#151) is not in mixedCase
Variable AbsToken._minTotal (ChuanJG.sol#152) is not in mixedCase
Variable AbsToken._receiveAddress (ChuanJG.sol#154) is not in mixedCase
Variable AbsToken._airdropLen (ChuanJG.sol#156) is not in mixedCase
Variable AbsToken._airdropAmount (ChuanJG.sol#157) is not in mixedCase
Variable AbsToken._removeLPFee (ChuanJG.sol#159) is not in mixedCase
Variable AbsToken._addLPFee (ChuanJG.sol#160) is not in mixedCase
Variable AbsToken._lpFeeReceiver (ChuanJG.sol#161) is not in mixedCase
Constant AbsToken._killBlock (ChuanJG.sol#163) 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 (ChuanJG.sol#446) is too similar to ISwapRouter.addLiquidity(address,address,uint256,uint256,uin
t256,uint256,address,uint256).amountBDesired (ChuanJG.sol#447)
Variable AbsToken.constructor(address,address,string,string,uint8,uint256,address,address,address,address,uint256).FundAddress2 (ChuanJG.sol#174) is too similar to AbsToken.constructor(address,address,string,s
tring,uint8,uint256,address,address,address,address,uint256).FundAddress3 (ChuanJG.sol#174)
Variable AbsToken.constructor(address,address,string,string,uint8,uint256,address,address,address,address,uint256).FundAddress2 (ChuanJG.sol#174) is too similar to AbsToken.fundAddress3 (ChuanJG.sol#111)
Variable AbsToken._buyFundFee (ChuanJG.sol#132) is too similar to AbsToken.setBuyFee(uint256,uint256,uint256,uint256,uint256).buyFundFee2 (ChuanJG.sol#595)
Variable AbsToken._buyFundFee (ChuanJG.sol#132) is too similar to AbsToken.setBuyFee(uint256,uint256,uint256,uint256,uint256,uint256).buyFundFee3 (ChuanJG.sol#595)
Variable AbsToken._buyFundFee2 (ChuanJG.sol#133) is too similar to AbsToken._buyFundFee3 (ChuanJG.sol#134)
Variable AbsToken._sellFundFee2 (ChuanJG.sol#140) is too similar to AbsToken._sellFundFee3 (ChuanJG.sol#141)
Variable AbsToken.setBuyFee(uint256,uint256,uint256,uint256,uint256,uint256).buyFundFee2 (ChuanJG.sol#595) is too similar to AbsToken.setBuyFee(uint256,uint256,uint256,uint256,uint256,uint256).buyFundFee3 (Chu
anJG.sol#595)
```

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- Start Trading (passed):

https://testnet.bscscan.com/tx/0xda8a7e0368a2c131b78a6bac2dbaa43679d799bba10dfb6d8606186086ef12ad

2- Set Fund Address (passed):

https://testnet.bscscan.com/tx/0x5a7c682de80fd52ddf8bbed390cd6198ee12c8df21c0fc07632f5f45cfbe1396

3- Set Fund Address2 (passed):

https://testnet.bscscan.com/tx/0x5758f94c76763554f0d6758e061205a8af05584ec91d2dc6373fecd1bca11936

4- Set Fund Address3 (passed):

https://testnet.bscscan.com/tx/0x7ecbcde82935c226e7656ef0a697a5a3653c63a0616a980543e4db0a6a8c1323

5- Set LP Fee Receiver (passed):

https://testnet.bscscan.com/tx/0x96c8c963544802bd420ffbda79f6badd35aac2e7f1fceb6d0819b30ceb526728

6- Set Receive Address (passed):

https://testnet.bscscan.com/tx/0x7b990a69bfe819a538b09872c6c62c601af905d1b2ab6381b2a6f109924b65e9

7- Set Sell Fee (passed):

https://testnet.bscscan.com/tx/0xa14657ea6fe6cef98a8cf3d807ab7cbb5d9d80500105e7a5b9fa0530418032a4

FUNCTIONAL TESTING

8- Set Buy Fee (passed):

https://testnet.bscscan.com/tx/0x6572042b5fcb80bfce84ee734341c581e6e02414bae7220ddab1f4933e4370b6

9- Set Transfer Fee (passed):

https://testnet.bscscan.com/tx/0x9ee9cd9fb51a53502dd44230d32bbf206729d7fb722dccc8f9a0e031c6cc3e57

10- Set Add LP Fee (passed):

https://testnet.bscscan.com/tx/0x9ee9cd9fb51a53502dd44230d32bbf206729d7fb722dccc8f9a0e031c6cc3e57

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	3
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Transfer Fees.

Severity: High

Function: setTransferFee

Status: Open

Overview:

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

```
function setTransferFee(uint256 fee) external onlyOwner {
    require(_transferFee < 2050, "fee too high");
   _transferFee = fee;
 }
```

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: startTrading

Status: Open

Overview:

The startTrading 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 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: Open

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 setFundAddress(address addr) external onlyOwner {
    fundAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress2(address addr) external onlyOwner {
    fundAddress2 = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress3(address addr) external onlyOwner {
    fundAddress3 = addr;
    _feeWhiteList[addr] = true;
  }
function setReceiveAddress(address addr) external onlyOwner {
    _receiveAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setLPFeeReceiver(address adr) external onlyOwner {
    _lpFeeReceiver = adr;
    _feeWhiteList[adr] = true;
  }
```

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 receiveAddress.

```
if (lpUsdt > 0) {
    _swapRouter.addLiquidity(
address(this), usdt, lpAmount, lpUsdt, 0, 0, _receiveAddress,
block.timestamp
    );
  }
```

Suggestion:

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

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 setFundAddress(address addr) external onlyOwner {
    fundAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress2(address addr) external onlyOwner {
    fundAddress2 = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress3(address addr) external onlyOwner {
    fundAddress3 = addr;
    _feeWhiteList[addr] = true;
  }
function setReceiveAddress(address addr) external onlyOwner {
    _receiveAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setLPFeeReceiver(address adr) external onlyOwner {
    _lpFeeReceiver = adr;
    _feeWhiteList[adr] = true;
  }
function setBuyFee(
uint256 buyDestroyFee, uint256 buyFundFee, uint256 buyFundFee2,
uint256 buyFundFee3,
uint256 lpDividendFee, uint256 lpFee
```

MANUAL TESTING

```
 ) external onlyOwner {
    _buyDestroyFee = buyDestroyFee;
    _buyFundFee = buyFundFee;
    _buyFundFee2 = buyFundFee2;
    _buyFundFee3 = buyFundFee3;
    _buyLPDividendFee = lpDividendFee;
    _buyLPFee = lpFee;
require(_buyDestroyFee + _buyFundFee + _buyFundFee2 + _buyFundFee3
+ _buyLPDividendFee + _buyLPFee < 2050, "fee too high");
  }
function setSellFee(
uint256 sellDestroyFee, uint256 sellFundFee, uint256 sellFundFee2,
uint256 sellFundFee3,
uint256 lpDividendFee, uint256 lpFee
  ) external onlyOwner {
    _sellDestroyFee = sellDestroyFee;
    _sellFundFee = sellFundFee;
    _sellFundFee2 = sellFundFee2;
    _sellFundFee3 = sellFundFee3;
    _sellLPDividendFee = lpDividendFee;
    _sellLPFee = lpFee;
require(_sellDestroyFee + _sellFundFee + _sellFundFee2 + _sellFundFee3
+ _sellLPDividendFee + _sellLPFee < 2050, "fee too high");
  }
function setTransferFee(uint256 fee) external onlyOwner {
require(_transferFee < 2050, "fee too high");
    _transferFee = fee;
  }
```

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Subject: Zero Check

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 setFundAddress2(address addr) external onlyOwner {
    fundAddress2 = addr;
    _feeWhiteList[addr] = true;
  }
function setFundAddress3(address addr) external onlyOwner {
    fundAddress3 = addr;
    _feeWhiteList[addr] = true;
  }
function setReceiveAddress(address addr) external onlyOwner {
    _receiveAddress = addr;
    _feeWhiteList[addr] = true;
  }
function setLPFeeReceiver(address adr) external onlyOwner {
    _lpFeeReceiver = adr;
    _feeWhiteList[adr] = true;
  }
```

MANUAL TESTING

Centralization – Local variable Shadowing

Severity: Low

Subject: Variable Shadowing

Status: Open

Overview:

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

Suggestion:

Rename the local variables that shadow another component.

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 ^0.8.18;

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.

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