auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Price AI

DATED : 23 JAN 23'

AUDIT SUMMARY

Project name – Price Ai

Date: 23 January , 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	0	0	0	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x5cf535b9776e5Ca01F3512228CaeF3d1ad7988B2

Token Information

Token Name : PriceAI

Token Symbol: PRICE

Decimals: 18

Token Supply: 500,000,000

Token Address:
0xb69edbD0527F448b650676C7085E15a7422791c5

Checksum:
f0e4c2f76c58916ec258f246851bea091d14d4247a2fc3e18694461b1816e13b

Owner:
0xB13D849AE23d306E0461311d8B5e04cBaB9b244A

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

- ✓ 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

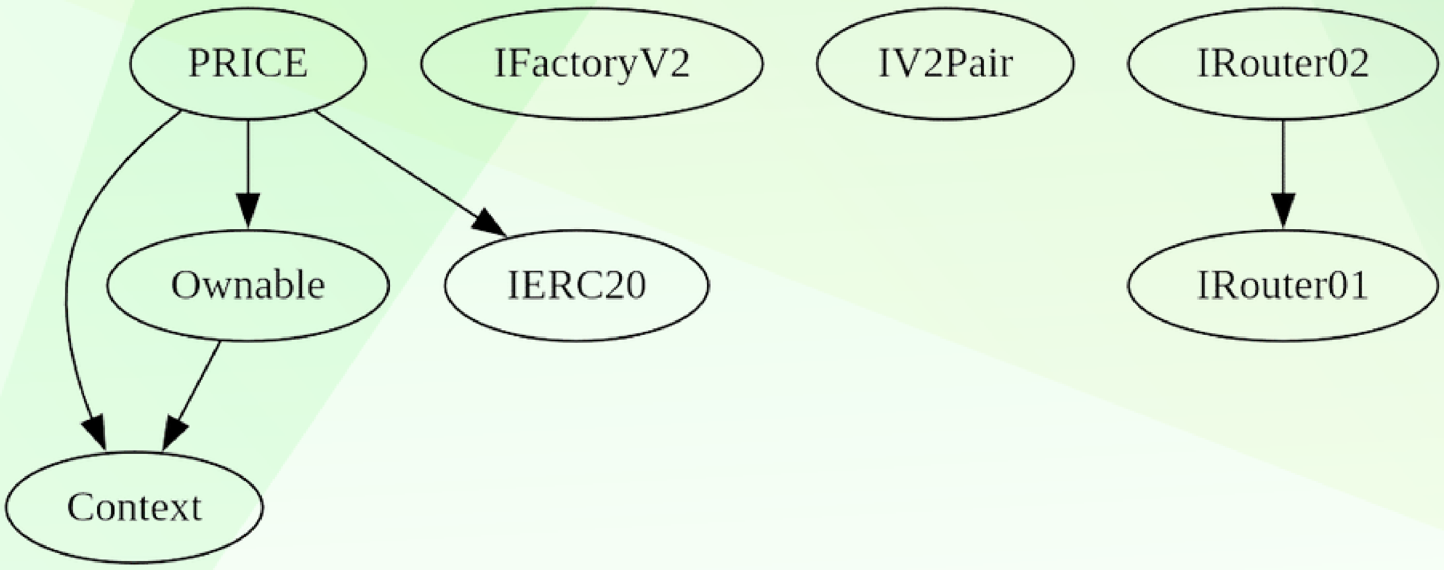
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change taxes more than 3% (3% buy and 3% sell, 0% transfer)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

```
| Contract | Type | Bases | | |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **Context** | Implementation | |||
| └ | <Constructor> | Public ❗️ | 🛑 |NO❗️ |
| └ | _msgSender | Internal 🔒 | | |
| └ | _msgData | Internal 🔒 | | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑 |NO❗️ |
| └ | owner | Public ❗️ | |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑 | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑 | |
||||||
| **IFactoryV2** | Interface | |||
| └ | getPair | External ❗️ | |NO❗️ |
| └ | createPair | External ❗️ | 🛑 |NO❗️ |
||||||
| **IV2Pair** | Interface | |||
| └ | factory | External ❗️ | |NO❗️ |
| └ | getReserves | External ❗️ | |NO❗️ |
| └ | sync | External ❗️ | 🛑 |NO❗️ |
||||||
| **IRouter01** | Interface | |||
| └ | factory | External ❗️ | |NO❗️ |
| └ | WETH | External ❗️ | |NO❗️ |
| └ | addLiquidityETH | External ❗️ | 💵 |NO❗️ |
| └ | addLiquidity | External ❗️ | 🛑 |NO❗️ |
| └ | swapExactETHForTokens | External ❗️ | 💵 |NO❗️ |
| └ | getAmountsOut | External ❗️ | |NO❗️ |
| └ | getAmountsIn | External ❗️ | |NO❗️ |
||||||
| **IRouter02** | Interface | IRouter01 |||
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑 |NO❗️ |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗️ | 💵 |NO❗️ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗️ | 🛑 |NO❗️ |
| └ | swapExactTokensForTokens | External ❗️ | 🛑 |NO❗️ |
||||||
| **IERC20** | Interface | |||
| └ | totalSupply | External ❗️ | |NO❗️ |
```

CONTRACT ASSESMENT

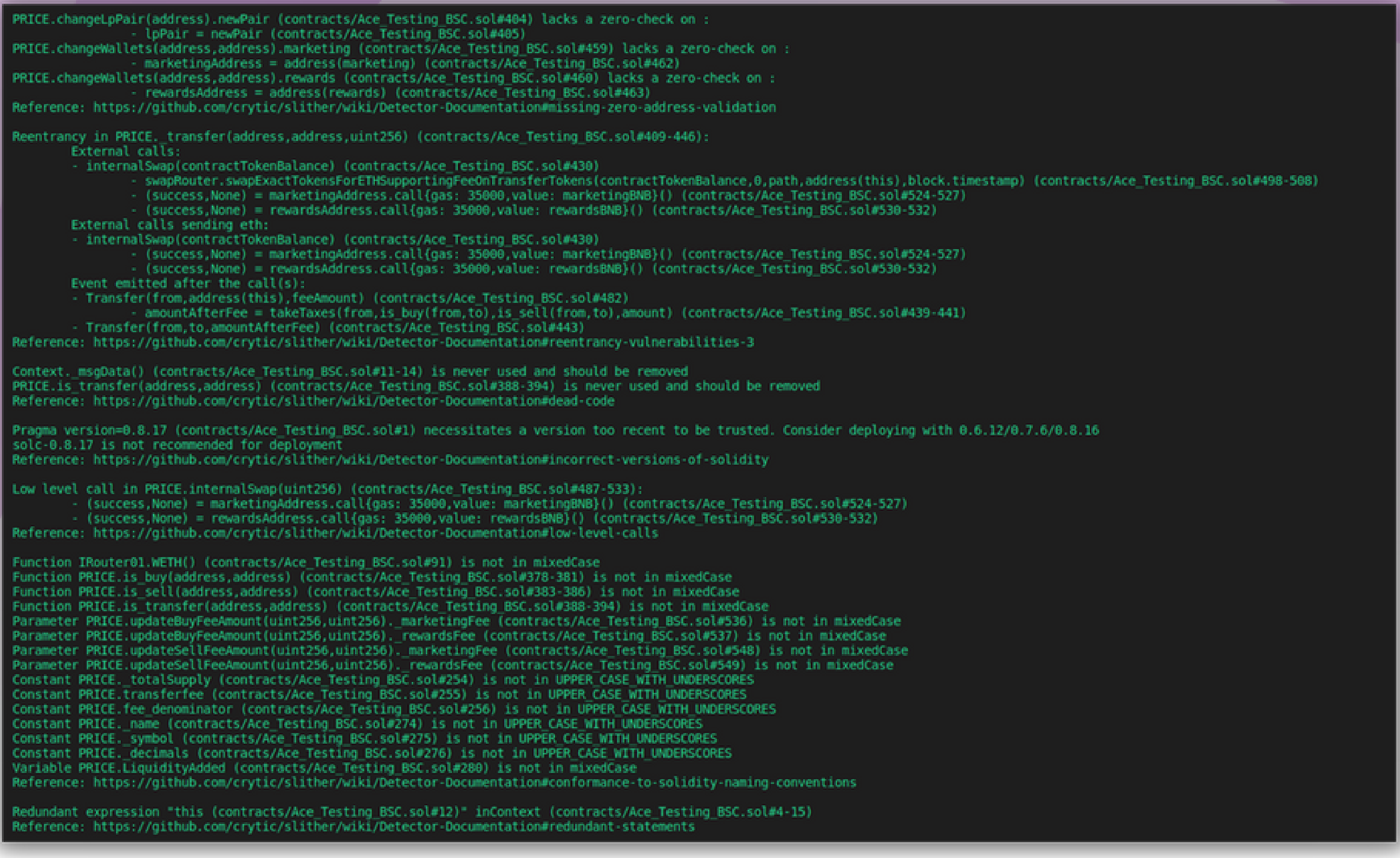
```
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **PRICE** | Implementation | Context, Ownable, IERC20 |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | balanceOf | Public ❗ |   |NO❗ |
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |
| └ | transfer | Public ❗ | 🛑  |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
| └ | isNoFeeWalelt | External ❗ |   |NO❗ |
| └ | setNoFeeWallet | Public ❗ | 🛑  | onlyOwner |
| └ | isLimitedAddress | Internal 🔒 |   | |
| └ | is_buy | Internal 🔒 |   | |
| └ | is_sell | Internal 🔒 |   | |
| └ | is_transfer | Internal 🔒 |   | |
| └ | canSwap | Internal 🔒 |   | |
| └ | changeLpPair | External ❗ | 🛑  | onlyOwner |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _basicTransfer | Internal 🔒 | 🛑  | |
| └ | changeWallets | External ❗ | 💵 | onlyOwner |
| └ | takeTaxes | Internal 🔒 | 🛑  | |
| └ | internalSwap | Internal 🔒 | 🛑  | inSwapFlag |
| └ | updateBuyFeeAmount | External ❗ | 🛑  | onlyOwner |
```

CONTRACT ASSESMENT

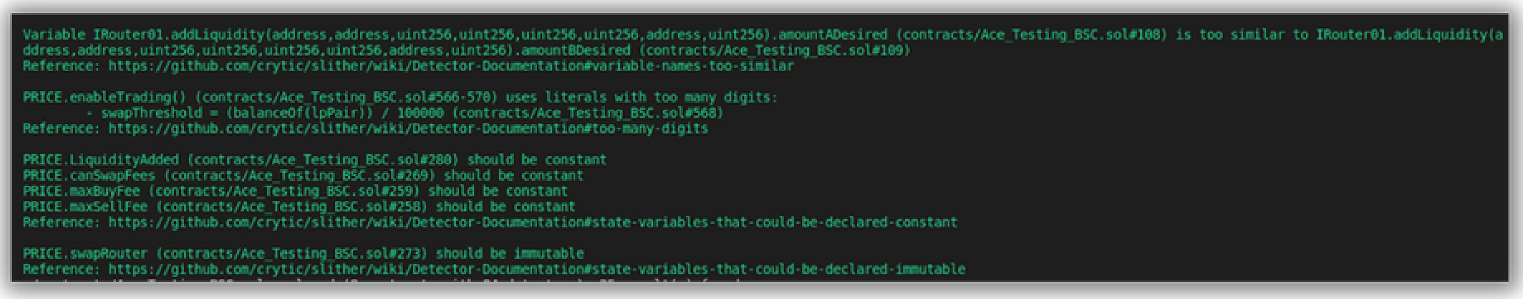
└	updateSellFeeAmount	External ❗	🛑	onlyOwner
└	setPresaleAddress	External ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner
└	rescueETH	External ❗	🛑	onlyOwner
└	rescueERC20	External ❗	🛑	onlyOwner

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
PRICE.changeLpPair(address).newPair (contracts/Ace_Testing_BSC.sol#404) lacks a zero-check on :
                - lpPair = newPair (contracts/Ace_Testing_BSC.sol#405)
PRICE.changeWallets(address,address).marketing (contracts/Ace_Testing_BSC.sol#459) lacks a zero-check on :
                - marketingAddress = address(marketing) (contracts/Ace_Testing_BSC.sol#462)
PRICE.changeWallets(address,address).rewards (contracts/Ace_Testing_BSC.sol#460) lacks a zero-check on :
                - rewardsAddress = address(rewards) (contracts/Ace_Testing_BSC.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in PRICE._transfer(address,address,uint256) (contracts/Ace_Testing_BSC.sol#409-446):
        External calls:
        - internalSwap(contractTokenBalance) (contracts/Ace_Testing_BSC.sol#430)
                - swapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(contractTokenBalance,0,path,address(this),block.timestamp) (contracts/Ace_Testing_BSC.sol#498-508)
                - (success,None) = marketingAddress.call{gas: 35000,value: marketingBNB}() (contracts/Ace_Testing_BSC.sol#524-527)
                - (success,None) = rewardsAddress.call{gas: 35000,value: rewardsBNB}() (contracts/Ace_Testing_BSC.sol#530-532)
        External calls sending eth:
        - internalSwap(contractTokenBalance) (contracts/Ace_Testing_BSC.sol#430)
                - (success,None) = marketingAddress.call{gas: 35000,value: marketingBNB}() (contracts/Ace_Testing_BSC.sol#524-527)
                - (success,None) = rewardsAddress.call{gas: 35000,value: rewardsBNB}() (contracts/Ace_Testing_BSC.sol#530-532)
        Event emitted after the call(s):
        - Transfer(from,address(this),feeAmount) (contracts/Ace_Testing_BSC.sol#482)
                - amountAfterFee = takeTaxes(from,is_buy(from,to),is_sell(from,to),amount) (contracts/Ace_Testing_BSC.sol#439-441)
        - Transfer(from,to,amountAfterFee) (contracts/Ace_Testing_BSC.sol#443)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Ace_Testing_BSC.sol#11-14) is never used and should be removed
PRICE.is_transfer(address,address) (contracts/Ace_Testing_BSC.sol#388-394) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.8.17 (contracts/Ace_Testing_BSC.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in PRICE.internalSwap(uint256) (contracts/Ace_Testing_BSC.sol#487-533):
        - (success,None) = marketingAddress.call{gas: 35000,value: marketingBNB}() (contracts/Ace_Testing_BSC.sol#524-527)
        - (success,None) = rewardsAddress.call{gas: 35000,value: rewardsBNB}() (contracts/Ace_Testing_BSC.sol#530-532)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter01.WETH() (contracts/Ace_Testing_BSC.sol#91) is not in mixedCase
Function PRICE.is_buy(address,address) (contracts/Ace_Testing_BSC.sol#378-381) is not in mixedCase
Function PRICE.is_sell(address,address) (contracts/Ace_Testing_BSC.sol#383-386) is not in mixedCase
Function PRICE.is_transfer(address,address) (contracts/Ace_Testing_BSC.sol#388-394) is not in mixedCase
Parameter PRICE.updateBuyFeeAmount(uint256,uint256)._marketingFee (contracts/Ace_Testing_BSC.sol#536) is not in mixedCase
Parameter PRICE.updateBuyFeeAmount(uint256,uint256)._rewardsFee (contracts/Ace_Testing_BSC.sol#537) is not in mixedCase
Parameter PRICE.updateSellFeeAmount(uint256,uint256)._marketingFee (contracts/Ace_Testing_BSC.sol#548) is not in mixedCase
Parameter PRICE.updateSellFeeAmount(uint256,uint256)._rewardsFee (contracts/Ace_Testing_BSC.sol#549) is not in mixedCase
Constant PRICE._totalSupply (contracts/Ace_Testing_BSC.sol#254) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE.transferfee (contracts/Ace_Testing_BSC.sol#255) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE.fee_denominator (contracts/Ace_Testing_BSC.sol#256) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._name (contracts/Ace_Testing_BSC.sol#274) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._symbol (contracts/Ace_Testing_BSC.sol#275) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._decimals (contracts/Ace_Testing_BSC.sol#276) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PRICE.LiquidityAdded (contracts/Ace_Testing_BSC.sol#280) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Ace_Testing_BSC.sol#12)" inContext (contracts/Ace_Testing_BSC.sol#4-15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```



```
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Ace_Testing_BSC.sol#108) is too similar to IRouter01.addLiquidity(a
ddress,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Ace_Testing_BSC.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PRICE.enableTrading() (contracts/Ace_Testing_BSC.sol#566-570) uses literals with too many digits:
        - swapThreshold = (balanceOf(lpPair)) / 100000 (contracts/Ace_Testing_BSC.sol#568)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PRICE.LiquidityAdded (contracts/Ace_Testing_BSC.sol#280) should be constant
PRICE.canSwapFees (contracts/Ace_Testing_BSC.sol#269) should be constant
PRICE.maxBuyFee (contracts/Ace_Testing_BSC.sol#259) should be constant
PRICE.maxSellFee (contracts/Ace_Testing_BSC.sol#258) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PRICE.swapRouter (contracts/Ace_Testing_BSC.sol#273) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => No issues found

FUNCTIONAL TESTING

0- Deploying (Passed):

https://testnet.bscscan.com/tx/0xfa979edeceba15c28dd1ee97e369fbdbdebee32a08c4a55ee829ea3007e5b322

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

https://testnet.bscscan.com/tx/0x6a544e3322e9241ee8cfce6d50a0823f83ede69acd3cf5ea1def78e87705d69f

no issue were found on adding liquidity.

2- Enabling trade for public (Passed):

https://testnet.bscscan.com/tx/0xb0a65ea6ae1b6dc1d71d95edc12a4034f59a0cfc814df82f54cd07f23028218f

3- Updating buy and sell taxes (2.99% each one) (Passed):

https://testnet.bscscan.com/tx/0x88fdc648e87381e88ff82d03832f6c06255594b66468b5b1af56d05be6052fb0

https://testnet.bscscan.com/tx/0x9fc801e442bab89b48006ee1a4431b955222e1f61dae5aa037f68e203c139384

4- Excluding deployer's wallet from taxes (to test taxes) (Passed):
https://testnet.bscscan.com/tx/0x2d5fd8b2c695c019a590fbbc4025b6fee9cc8c9ede445058dd09295c74a6cdb7

FUNCTIONAL TESTING

5- Buying from a non-excluded wallet (Passed):

https://testnet.bscscan.com/tx/0x471c017ddc3be5fcb31274f29fb4d7aa50c8660e377eac77eae2aa74b8349075

2.99% tax on buys

6- Selling from a non-excluded wallet (Passed):

https://testnet.bscscan.com/tx/0x7e13d8e2d1953d6e1b8668283ea298da5beb78d4c809d65140a65713a4a9676c

2.99% tax on sells

7- Swap & liquifiy (Passed):

taxes were swapped to BNB and sent to marketing & rewards wallets.

https://testnet.bscscan.com/tx/0x7e13d8e2d1953d6e1b8668283ea298da5beb78d4c809d65140a65713a4a9676c

MANUAL TESTING

Optimzation and Suggestions

Gas Optimizations:

- approve router for one time at constructor and avoid approving at intrenalSwap
- define this variables as constant:

maxSellFee

maxBuyFee

Suggestions:

- create a function to be able to change swap thershold

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