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: Failed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	1	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- Goerli:
all tests were done on Goerli 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 goerli

https://goerli.etherscan.io/address/0x601ea5876E6a3a954A5D2607Bed8534EfD25ed45

Testnet BUSD contract (used to test the contract):

https://goerli.etherscan.io/token/0xc6a19abd1818c0b60f208f38d88a2abd5a79c32e

Token Information

Token Name : PriceA

Token Symbol: Price

Decimals: 18

Token Supply: 500,000,000

Token Address:
0x0DeCf94c4549A3E3e1B444bB9c229cC1846B2F74

Checksum:
f0e4c2f76c58916ec258f246851bea091d14d4247a2fc3e18694461b1816e13b

Deployer:
0x9f282ebcdcd9c2604b79468E182f174848fFf195

Owner:
0x0738aEe3c16736b993477227025C861CFAf48A0F

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

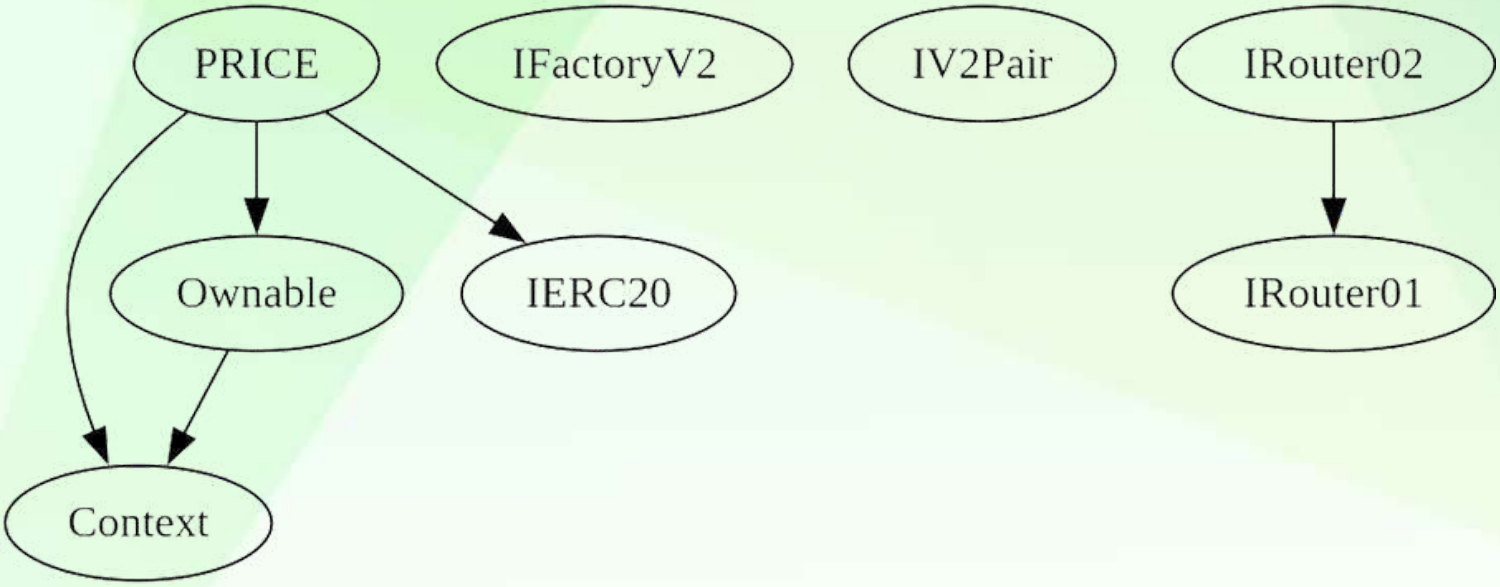
CLASSIFICATION OF RISK

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	10
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change taxes (1% 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 ❗

CONTRACT ASSESMENT

| └ | 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 ❗ |

| └ | 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, I ERC20 |||

| └ | totalSupply | External ❗ | | NO ❗ |

| └ | decimals | External ❗ | | NO ❗ |

| └ | symbol | External ❗ | | NO ❗ |

| └ | name | External ❗ | | NO ❗ |

CONTRACT ASSESMENT

| └ | 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 ❗ |

| └ | isNoFeeWaleIt | 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 |

| └ | updateSellFeeAmount | External ❗ | 🛑 | onlyOwner |

| └ | setPresaleAddress | External ❗ | 🛑 | onlyOwner |

| └ | enableTrading | External ❗ | 🛑 | onlyOwner |

| └ | rescueETH | External ❗ | 🛑 | onlyOwner |

CONTRACT ASSESMENT

| └ | rescueERC20 | External ❗ | 🛑 | onlyOwner |

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
PRICE.changeWallets(address,address).rewards (contracts/token.sol#332) lacks a zero-check on :
                - rewardsAddress = address(rewards) (contracts/token.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in PRICE._transfer(address,address,uint256) (contracts/token.sol#293-323):
        External calls:
        - internalSwap(contractTokenBalance) (contracts/token.sol#309)
                - swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalance,0,path,address(this),block.timestamp) (contracts/token.sol#371-379)
                - IERC20(busdToken).transferFrom(address(this),marketingAddress,marketingBUSD) (contracts/token.sol#387)
                - IERC20(busdToken).transferFrom(address(this),rewardsAddress,rewardsBUSD) (contracts/token.sol#388)
        Event emitted after the call(s):
        - Transfer(from,address(this),feeAmount) (contracts/token.sol#348)
                - amountAfterFee = takeTaxes(from,is_buy(from,to),is_sell(from,to),amount) (contracts/token.sol#317)
        - Transfer(from,to,amountAfterFee) (contracts/token.sol#319)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/token.sol#16-19) is never used and should be removed
PRICE.is_transfer(address,address) (contracts/token.sol#277-280) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.8.17 (contracts/token.sol#5) 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

Function IRouter01.WETH() (contracts/token.sol#71) is not in mixedCase
Function PRICE.is_buy(address,address) (contracts/token.sol#267-270) is not in mixedCase
Function PRICE.is_sell(address,address) (contracts/token.sol#272-275) is not in mixedCase
Function PRICE.is_transfer(address,address) (contracts/token.sol#277-280) is not in mixedCase
Parameter PRICE.updateBuyFeeAmount(uint256,uint256)._marketingFee (contracts/token.sol#392) is not in mixedCase
Parameter PRICE.updateBuyFeeAmount(uint256,uint256)._rewardsFee (contracts/token.sol#392) is not in mixedCase
Parameter PRICE.updateSellFeeAmount(uint256,uint256)._marketingFee (contracts/token.sol#398) is not in mixedCase
Parameter PRICE.updateSellFeeAmount(uint256,uint256)._rewardsFee (contracts/token.sol#398) is not in mixedCase
Constant PRICE._totalSupply (contracts/token.sol#168) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE.transferfee (contracts/token.sol#169) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE.fee_denominator (contracts/token.sol#170) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._name (contracts/token.sol#187) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._symbol (contracts/token.sol#188) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PRICE._decimals (contracts/token.sol#189) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PRICE.LiquidityAdded (contracts/token.sol#194) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/token.sol#17)" inContext (contracts/token.sol#8-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/token.sol#83) is too similar to IRouter01.addLiquid
ity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/token.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PRICE.enableTrading() (contracts/token.sol#411-415) uses literals with too many digits:
        - swapThreshold = (balanceOf(lpPair)) / 100000 (contracts/token.sol#413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PRICE.LiquidityAdded (contracts/token.sol#194) should be constant
PRICE.busdToken (contracts/token.sol#191) should be constant
PRICE.maxBuyFee (contracts/token.sol#173) should be constant
PRICE.maxSellFee (contracts/token.sol#172) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PRICE.swapRouter (contracts/token.sol#186) should be immutable
```

Result => static analysis has been performed using Slither, there was not any issues found except some minimal suggestions for improvement which are mentioned at above picture (slither output)

FUNCTIONAL TESTING

Functionality tests for ERC20 tokens includes:
- adding liquidity
- buying / selling /transferring (for non-excluded wallets)
- checking tax conversions, tax destinations
- checking auto liquidity

1- Adding Liquidity:

liquidity added on Uniswap v2:

https://goerli.etherscan.io/tx/0xfbd922365a21c8bcb5d79dbe60d752934c853f49039bb141f5dd7e6c6d942d94

no issue were found on adding liquidity.

2- Buying from a non-excluded wallet:

https://goerli.etherscan.io/tx/0xadb71ceaed53aae67349334da120e02e3a383e78d846f5079a60adad6a0f528e

3% tax on buy, transferred to contract (mainnet contract has 0 buy tax, we increased it here to test buy tax)

3- Selling from a non-excluded wallet (FAILED)

could not sell, please refer to issues section

FUNCTIONAL TESTING

4- Swap & liquifiy (FAILED)

accumulated taxes did not convert to BUSD, this is due to a critical issue in the contract which is mentioned in the issues section.

MANUAL TESTING

Critical Risk Findings:

1) Contract is not able to send BUSD tokens to marketing and reward wallet due to lack of allowance on behalf of itself.
Since allowance of contract to spend BUSD tokens on behalf of itself is zero:

BUSD_Contract.allowance(Price_contract, Price_contract) == 0
then sell transactions will be reverted at “internalSwap” function at this lines:

```
IERC20(busdToken).transferFrom(
        address(this),
        marketingAddress,
        marketingBUSD);
IERC20(busdToken).transferFrom(
        address(this),
        rewardsAddress,
        rewardsBUSD);
```

Suggestion:
there is 2 ways to solve this issue:
1- use transfer instead of transferFrom for sending BUSD out of contract
2- approve contract to spend BUSD on behalf of itself:
“IERC20(busdToken).approve(address(this), ~uint256(0))”

MANUAL TESTING

Optimzation

Gas Optimizations:

- you can approve swapRouter at constructor and ignore this operations at “internalSwap” to reduce overall buy and sell gas

.

```
if (_allowances[address(this)][address(swapRouter)] != type(uint256).max) {
    _allowances[address(this)][address(swapRouter)] = type(uint256).max;
}

if (_allowances[busdToken][address(this)] != type(uint256).max) {
    _allowances[busdToken][address(this)] = type(uint256).max;
}
```

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