auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Kong Inu

DATED : 17 APRIL 23'

AUDIT SUMMARY

Project name – Kong Inu

Date:17 April, 2023

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

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	0	0
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 Testnet 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 Link: Contract has been tested on binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/address/0x3E52AEb787C10f9b275E2B361fEEe79b6bF5DceF#code

Token Information

Token Name : Kong Inu

Token Symbol: KONG

Decimals: 9

Token Supply: 210,000,000,000,000,000

Token Address:
0xAC1Cd89092e04c87f13D6f80fb9A9a506468678e

Checksum:
7b0d6058387825c26af841b8913f24278a52984c

Owner:
0xC344EA9B5ada0E9489626b5Da271d5f6E8103B7f
(at time of audit)

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 10%

Fees Privilege: None

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: updating liquidity threshold - excluding from fees - including in fees - including in rewards - excluding from rewards

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

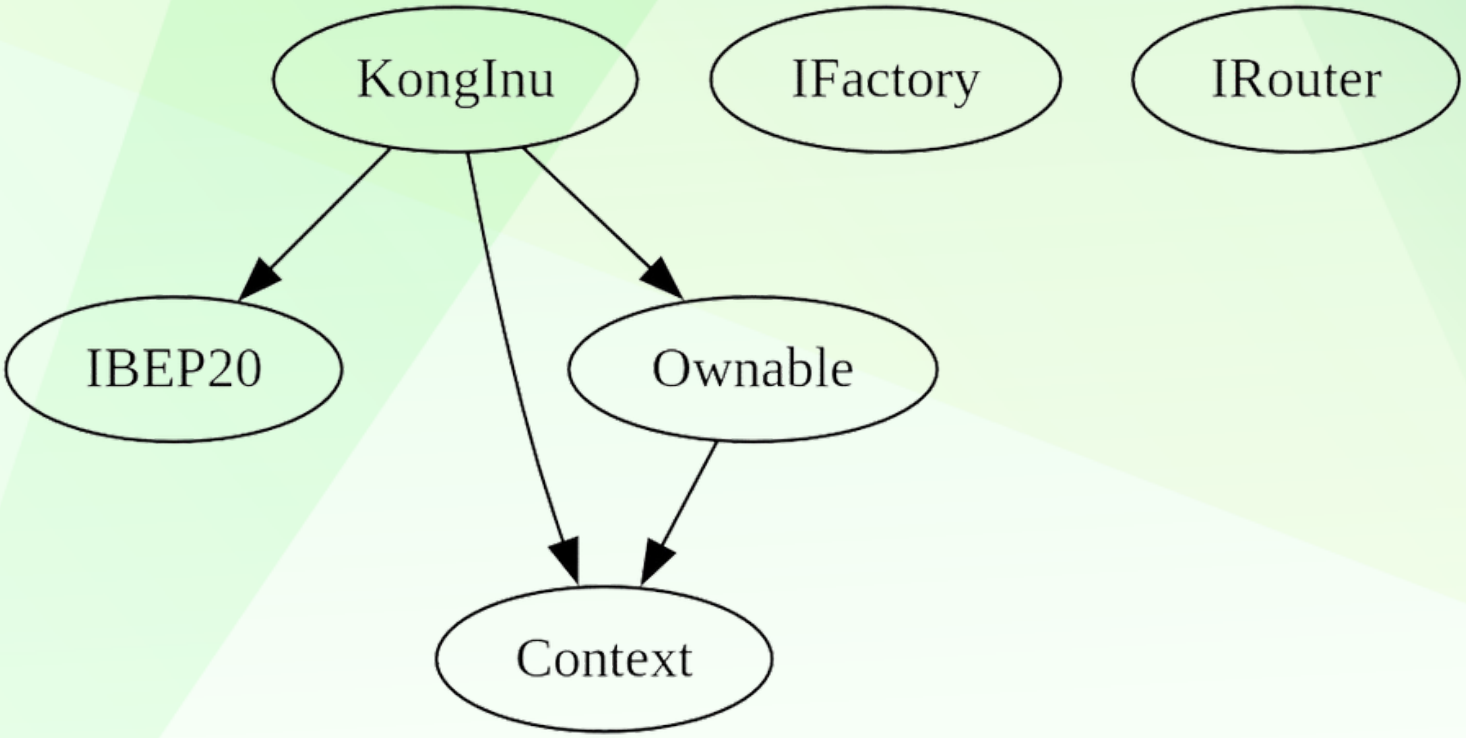
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to modify buy/sell/transfer fees (10% each)
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- 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**
IBEP20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
Address	Library			
└	sendValue	Internal 🔒	🛑	
KongInu	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗

CONTRACT ASSESMENT

└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	isExcludedFromReward	Public ❗		NO ❗
└	reflectionFromToken	Public ❗		NO ❗
└	EnableTrading	External ❗	🛑	onlyOwner
└	updatedeadline	External ❗	🛑	onlyOwner
└	tokenFromReflection	Public ❗		NO ❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO ❗
└	_reflectRfi	Private 🔐	🛑	
└	_takeLiquidity	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	_takeOps	Private 🔐	🛑	
└	_takeDev	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues1	Private 🔐		
└	_getRValues2	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateOpsWallet	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Token Distribution

It should be noted that the owner currently holds 100% of the total supply. However, information about the distribution of these tokens is not available, and it is recommended that investors exercise caution when considering this aspect.

STATIC ANALYSIS

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

KongInu.includeInReward(address) (contracts/Token.sol#406-417) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#50-53) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

KongInu._rTotal (contracts/Token.sol#169) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#11) 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

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#130-141):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#106) is not in mixedCase
Struct KongInu.valuesFromGetValues (contracts/Token.sol#206-220) is not in CapWords
Function KongInu.EnableTrading() (contracts/Token.sol#372-377) is not in mixedCase
Parameter KongInu.updatedeadline(uint256)._deadline (contracts/Token.sol#379) is not in mixedCase
Parameter KongInu.updateSwapEnabled(bool)._enabled (contracts/Token.sol#795) is not in mixedCase
Parameter KongInu.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#807) is not in mixedCase
Parameter KongInu.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#808) is not in mixedCase
Parameter KongInu.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#809) is not in mixedCase
Constant KongInu._decimals (contracts/Token.sol#165) is not in UPPER_CASE_WITH_UNDERSCORES
Variable KongInu.genesis_block (contracts/Token.sol#173) is not in mixedCase
Constant KongInu._name (contracts/Token.sol#181) is not in UPPER_CASE_WITH_UNDERSCORES
Constant KongInu._symbol (contracts/Token.sol#182) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#51)" inContext (contracts/Token.sol#45-54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

KongInu._lastSell (contracts/Token.sol#160) is never used in KongInu (contracts/Token.sol#144-819)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

KongInu._tTotal (contracts/Token.sol#168) should be constant
KongInu.deadWallet (contracts/Token.sol#176) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

KongInu.pair (contracts/Token.sol#163) should be immutable
KongInu.router (contracts/Token.sol#162) 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):
https://testnet.bscscan.com/tx/0xe796ffa88e06cfb489e5438c2cdd63a5849f37fac3df5af6dab906af5adee7be

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xa57f2e06f169dbc968282d4a0c4cff963f80d4d7b8e4129fbc283030c238f85a

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x530562a01d4302bee0c3d63baac9a8d2c1471260f3f21affb5d897328ea34501

4- Transferring when excluded from fees(0% tax) (passed):
https://testnet.bscscan.com/tx/0x8efbe2b7093e23aebe65070c88185f3158f6f62f2488bb01159ecd280e6d17a6

5- Buying when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0xb6c23ff15b98fc49951a7a60cf5e0ce08529a0dd7de6d3026e82bb1b9a491e00

6- Selling when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0x33b1bdd799b37188210cf453b30edc1428bb79d7e8b0bac1ec10b97baf492f7d

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0x07fa8f110f42be3a4a8566a28084962a03aef12145238a49737a99ef4c831b8d

8- Internal swap (passed):
All fee wallets received BNB
https://testnet.bscscan.com/address/0xe0a8b9318c60dcd3a533c462a25ae2ab453fc302#internaltx

MANUAL TESTING

Logical - Setting swap threshold to 0

Severity: High

Function: updateSwapTokensAtAmount

Lines: 787

Status: Not Resolved

Overview:

setting swap threshold to 0 can disable sells if contract balance is more than threshold.

```
  function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
      amount <= 42e14,
      "Cannot set swap threshold amount higher than 1% of tokens"
    );
    swapTokensAtAmount = amount * 10 ** _decimals;
  }
```

Recommendation:

ensure that swap threshold can not be zero.

Example:

```
  function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
      amount <= 42e14,
      "Cannot set swap threshold amount higher than 1% of tokens"
    );
    require(
      amount > 0,
"Cannot set swap threshold amount to 0"
    );
    swapTokensAtAmount = amount * 10 ** _decimals;
  }
```

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