auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

CHIBAINU

DATED :04 Jan, 2024

AUDIT SUMMARY

Project name – CHIBAINU

Date: 04 Jan, 2024

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	1	0	2
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/0x115544F9caE4ad469440D36daf0c004D31BDc336#code

Token Information

Token Address:
0xb2A3f1DF14b1aF76Ad6d53f4AD5fb51Bfde44D60

Name: CHIBAINU

Symbol: CHIBA

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner:
0xF4436AD72717ac3d963d75273baf984dA9a53F2A

Deployer:
0x4659d1a6eee90b5f108c5e60a13fe1fd32161d6d

Token Supply: 300,000,000

Checksum: Ade3cef7c2c788bc03532d7342fc9ghf

Testnet:
https://testnet.bscscan.com/address/0x115544F9caE4ad469440D36daf0c004D31BDc336#code

TOKEN OVERVIEW

Buy Fee: 5%

Sell Fee: 5%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

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

- 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

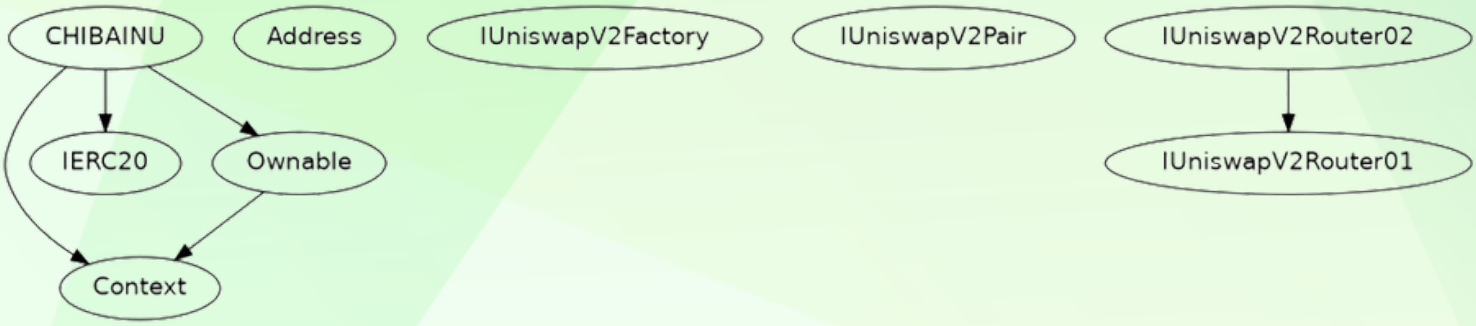
CLASSIFICATION OF RISK

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	1
Low-Risk	0
Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Exclude/Include wallets from fees.
- The owner can set the token to swap.
- The owner can set a marketing wallet address.

STATIC ANALYSIS

```
INFO:Detectors:
CHIBAINU.transferToAddressETH(address,uint256) (CHIBAINU.sol#745-752) uses a dangerous strict equality:
        - amount == 0 (CHIBAINU.sol#749)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
CHIBAINU.constructor().currentRouter (CHIBAINU.sol#551) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CHIBAINU.allowance(address,address)._owner (CHIBAINU.sol#583) shadows:
        - Ownable._owner (CHIBAINU.sol#197) (state variable)
CHIBAINU._approve(address,address,uint256)._owner (CHIBAINU.sol#631) shadows:
        - Ownable._owner (CHIBAINU.sol#197) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in CHIBAINU._transfer(address,address,uint256) (CHIBAINU.sol#637-675):
        External calls:
        - swapAndLiquify() (CHIBAINU.sol#657)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (CHIBAINU.sol#688-694)
        External calls sending eth:
        - swapAndLiquify() (CHIBAINU.sol#657)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
        State variables written after the call(s):
        - marketingTokensCollected += fee (CHIBAINU.sol#671)
        - totalMarketingTokensCollected += fee (CHIBAINU.sol#672)
Reentrancy in CHIBAINU.swapAndLiquify() (CHIBAINU.sol#676-682):
        External calls:
        - swapTokensForEth(totalTokens) (CHIBAINU.sol#678)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (CHIBAINU.sol#688-694)
        - transferToAddressETH(marketingWallet,ethBalance) (CHIBAINU.sol#680)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
        External calls sending eth:
        - transferToAddressETH(marketingWallet,ethBalance) (CHIBAINU.sol#680)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
        State variables written after the call(s):
        - marketingTokensCollected = 0 (CHIBAINU.sol#681)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CHIBAINU._transfer(address,address,uint256) (CHIBAINU.sol#637-675):
        External calls:
        - swapAndLiquify() (CHIBAINU.sol#657)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (CHIBAINU.sol#688-694)
        External calls sending eth:
        - swapAndLiquify() (CHIBAINU.sol#657)
                - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (CHIBAINU.sol#704)
                - _tokenTransfer(from,address(this),fee) (CHIBAINU.sol#670)
```

STATIC ANALYSIS

```
INFO:Detectors:
Address._revert(bytes,string) (CHIBAINU.sol#182-194) uses assembly
        - INLINE ASM (CHIBAINU.sol#187-190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
CHIBAINU.excludeFromFee(address) (CHIBAINU.sol#709-716) compares to a boolean constant:
        -require(bool,string)(_isExcludedFromFee[account] != true,The wallet is already excluded!) (CHIBAINU.sol#710-713)
CHIBAINU.includeInFee(address) (CHIBAINU.sol#717-724) compares to a boolean constant:
        -require(bool,string)(_isExcludedFromFee[account] != false,The wallet is already included!) (CHIBAINU.sol#718-721)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address._revert(bytes,string) (CHIBAINU.sol#182-194) is never used and should be removed
Address.functionCall(address,bytes) (CHIBAINU.sol#53-64) is never used and should be removed
Address.functionCall(address,bytes,string) (CHIBAINU.sol#65-71) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CHIBAINU.sol#72-84) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (CHIBAINU.sol#85-105) is never used and should be removed
Address.functionDelegateCall(address,bytes) (CHIBAINU.sol#131-141) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (CHIBAINU.sol#142-155) is never used and should be removed
Address.functionStaticCall(address,bytes) (CHIBAINU.sol#106-116) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (CHIBAINU.sol#117-130) is never used and should be removed
Address.isContract(address) (CHIBAINU.sol#39-41) is never used and should be removed
Address.sendValue(address,uint256) (CHIBAINU.sol#42-52) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (CHIBAINU.sol#171-181) is never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (CHIBAINU.sol#156-170) is never used and should be removed
CHIBAINU.swapETHForTokens(uint256) (CHIBAINU.sol#755-768) is never used and should be removed
Context._msgData() (CHIBAINU.sol#10-12) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.19 (CHIBAINU.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (CHIBAINU.sol#42-52):
        - (success) = recipient.call{value: amount}() (CHIBAINU.sol#47)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CHIBAINU.sol#85-105):
        - (success,returndata) = target.call{value: value}(data) (CHIBAINU.sol#95-97)
Low level call in Address.functionStaticCall(address,bytes,string) (CHIBAINU.sol#117-130):
        - (success,returndata) = target.staticcall(data) (CHIBAINU.sol#122)
Low level call in Address.functionDelegateCall(address,bytes,string) (CHIBAINU.sol#142-155):
        - (success,returndata) = target.delegatecall(data) (CHIBAINU.sol#147)
Low level call in CHIBAINU.transferToAddressETH(address,uint256) (CHIBAINU.sol#745-752):
        - (succ) = recipient.call{value: amount}() (CHIBAINU.sol#750)
Low level call in CHIBAINU.recoverETHfromContract() (CHIBAINU.sol#769-775):
        - (succ) = address(marketingWallet).call{value: ethBalance}() (CHIBAINU.sol#771)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (CHIBAINU.sol#276) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (CHIBAINU.sol#277) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (CHIBAINU.sol#303) is not in mixedCase
Function IUniswapV2Router01.WETH() (CHIBAINU.sol#329) is not in mixedCase
Parameter CHIBAINU.allowance(address,address)._owner (CHIBAINU.sol#583) is not in mixedCase
Parameter CHIBAINU.setTokensToSwap(uint256)._minimumTokensBeforeSwap (CHIBAINU.sol#726) is not in mixedCase
Parameter CHIBAINU.setSwapAndLiquifyEnabled(bool)._enabled (CHIBAINU.sol#735) is not in mixedCase
Parameter CHIBAINU.setMarketingWallet(address)._marketingWallet (CHIBAINU.sol#740) is not in mixedCase
Parameter CHIBAINU.recoverTokensFromContract(address,uint256)._tokenAddress (CHIBAINU.sol#777) is not in mixedCase
Parameter CHIBAINU.recoverTokensFromContract(address,uint256)._amount (CHIBAINU.sol#778) is not in mixedCase
Variable CHIBAINU.WETH (CHIBAINU.sol#541) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (CHIBAINU.sol#333) is too similar to IUniswapV2Router01.addLiquidity(address,address,uin
t256,uint256,uint256,uint256,address,uint256).amountBDesired (CHIBAINU.sol#334)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
CHIBAINU._tTotal (CHIBAINU.sol#529) should be constant
CHIBAINU.buyFee (CHIBAINU.sol#533) should be constant
CHIBAINU.sellFee (CHIBAINU.sol#535) should be constant
CHIBAINU.transferFee (CHIBAINU.sol#534) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:CHIBAINU.sol analyzed (9 contracts with 93 detectors), 59 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x814484f8f8799346f6ecce9132da8ad35e22fb96634a4b4d02308ea570607ca0

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0xa7a2a7b6839171af753f7d73cf738a73d0bbf287bd92606f1db1c1ca4d8e8903

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0x6224cb85df9adbc3c7d4a73325ffc02a1b9433b88d0be564762e9b0ccff4090a

4- Set Marketing Wallet (passed):

https://testnet.bscscan.com/tx/0x6c39226ad26a634191e4de083f61c653aec277ec45c8d98fdcf734b2329a79d3

MANUAL TESTING

Centralization – Missing Require Check.
Severity: Medium
Function: SetMarketingWallet
Status: Open

Overview:
The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set 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 setMarketingWallet(address _marketingWallet) external onlyOwner {
require(_marketingWallet != address(0), "setmarketingWallet: ZERO");
    marketingWallet = payable(_marketingWallet);
emit UpdateMarketingWallet(marketingWallet);
  }
```

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

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. Though it is the best practice to avoid them.

```
function _msgData() internal view virtual returns (bytes calldata)
{
return msg.data;
  }
event AuditLog(string, address);
event UpdateStakingWallet(address);
event UpdateBuyFee(uint256);
event UpdateSellFee(uint256);
event UpdateTransferFee(uint256);
event UpdateDistribution(uint256, uint256);
event TradingStarted(bool);
```

MANUAL TESTING

Optimization

Severity: Informational

Subject: uint256

Status: Open

Overview:
Use uit256 instead of uint. uint is an alias for uint256 and is not recommended for use. The variable size should be clarified, as this can cause issues when encoding data with selectors if the alias is mistakenly used within the signature string.

```
uint public buyFee = 5;
```

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