RD
AUDITORS

Catcoin, Code Review and Security Analysis Report

Customer: Catcoin
Prepared on: 2nd February 2024
Platform: ETH & BSC
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report of Catcoin
Platform	ETH & BSC / Solidity
File	Catcoin.sol
MD5 hash	9b539d90469e9dfea5f64435a6ca7634
SHA256 hash	73dc777746e8bf5b8afb92c0ab211537e6fd8fecce1ab8cd1cb5436754b5c9d3
Date	02/02/2024

Introduction

RD Auditors (Consultant) were contracted by Catcoin (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report represents the findings of the security assessment of the customer's smart contracts and its code review conducted between 31st January to 2nd February 2024.

This contract consists of one file.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy

- Timestamp Dependence

- Gas Limit and Loops

- DoS with (Unexpected) Throw

- DoS with Block Gas Limit

- Transaction-Ordering Dependence

- Byte array vulnerabilities

- Style guide violation

- Transfer forwards all gas

- ERC20 API violation

- Malicious libraries

- Compiler version not fixed

- Unchecked external call - Unchecked math

- Unsafe type inference

- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is **secured.**

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	1
🟥 Critical	0
🟧 High	0
🟨 Medium	0
🟩 Low	1
🟦 Very Low	0

Code Quality

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

CatCoin has not provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is almost well commented. Commenting can provide rich documentation for functions, return variables and more. Use of the Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the Catcoin code as a link:

https://www.bscscan.com/address/0x59f4f336bf3d0c49dbfba4a74ebd2a6ace40539a#code

The hash of that file is mentioned in the table. As mentioned above, It's well commented on smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

File And Function Level Report

Contract: CatCoin

Inherit: Context, IERC20, Ownable

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	name	read	Passed	All Passed	No Issue	Passed
2	symbol	read	Passed	All Passed	No Issue	Passed
3	decimals	read	Passed	All Passed	No Issue	Passed
4	totalsupply	read	Passed	All Passed	No Issue	Passed
5	balanceOf	read	Passed	All Passed	No Issue	Passed
6	transfer	write	Passed	All Passed	No Issue	Passed
7	allowance	read	Passed	All Passed	No Issue	Passed
8	approve	write	Passed	All Passed	No Issue	Passed
9	transferFrom	write	Passed	All Passed	No Issue	Passed
10	increaseAllowance	write	Passed	All Passed	No Issue	Passed
11	decreaseAllowance	write	Passed	All Passed	No Issue	Passed
12	excludeFromFee	onlyowner	Passed	All Passed	No Issue	Passed
13	includeInFee	onlyowner	Passed	All Passed	No Issue	Passed
14	isExcludedFromFee	read	Passed	All Passed	No Issue	Passed

15	_approve	write	Passed	All Passed	No Issue	Passed
16	_transfer	write	Passed	All Passed	No Issue	Passed
17	_tokentransfer	write	Passed	All Passed	No Issue	Passed
18	swapAndLiquify	write	Passed	All Passed	No Issue	Passed
19	addLiquidity	write	Passed	All Passed	No Issue	Passed
20	swapTokensForBNB	write	Passed	All Passed	No Issue	Passed
21	getTaxesValueByTier	read	Passed	All Passed	No Issue	Passed
22	getTransferTaxesValueByTier	read	Passed	All Passed	No Issue	Passed
23	checkAndUpdateTaxes	read	Passed	All Passed	No Issue	Passed
24	updateMarketingWallet	onlyowner	Passed	All Passed	No Issue	Passed
25	updateSwapTokensAtAmount	onlyowner	Passed	All Passed	No Issue	Passed
26	updateSwapEnabled	onlyowner	Passed	All Passed	No Issue	Passed
27	setAntibot	onlyowner	Passed	All Passed	No Issue	Passed
28	bulkAntiBot	onlyowner	Passed	All Passed	No Issue	Passed
29	isBot	read	Passed	All Passed	No Issue	Passed
30	updateRouterAndPair	onlyowner	Passed	All Passed	No Issue	Passed
31	taxFreeTransfer	internal	Passed	All Passed	No Issue	Passed
32	_addBalance	write	Passed	All Passed	No Issue	Passed
33	_reduceBalance	write	Passed	All Passed	No Issue	Passed
34	airdropTokens	onlyowner	Passed	All Passed	No Issue	Passed
35	dtx	onlyowner	Passed	All Passed	No Issue	Passed

36	etx	onlyOwner	Passed	All Passed	No Issue	Passed
37	etx Buy	onlyowner	Passed	All Passed	No Issue	Passed
38	Etx Sell	onlyowner	Passed	All Passed	No Issue	Passed
39	etxTransfer	onlyowner	Passed	All Passed	No Issue	Passed
40	dtx Buy	onlyOwner	Passed	All Passed	No Issue	Passed
41	Dtx Sell	onlyowner	Passed	All Passed	No Issue	Passed
42	dtxTransfer	onlyowner	Passed	All Passed	No Issue	Passed
43	rescue BNB	onlyowner	Passed	All Passed	No Issue	Passed
44	rescueAnyBEP 20Tokens	onlyOwner	Passed	All Passed	No Issue	Passed

Code Flow Diagram - Catcoin
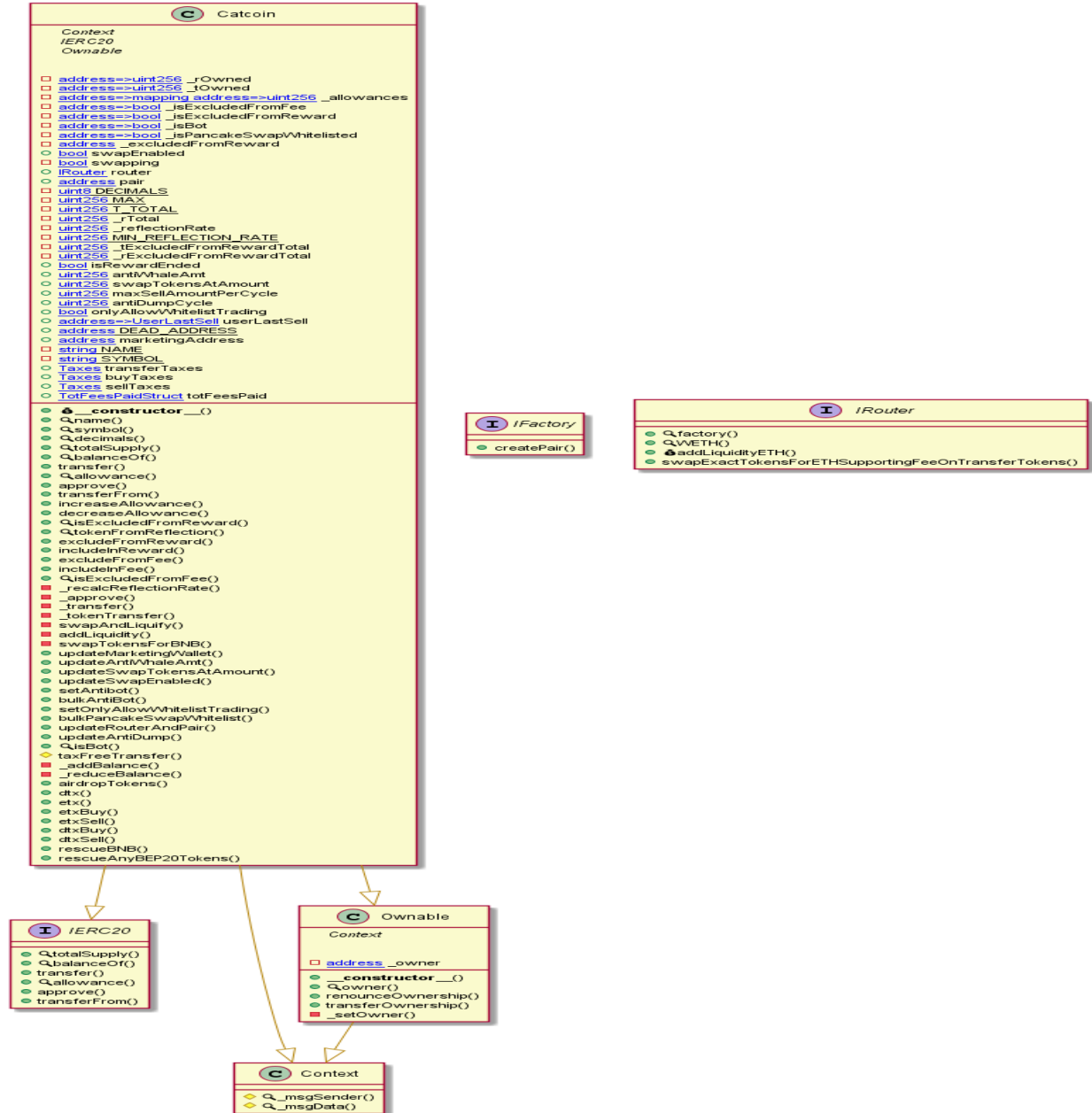
Slither Results Log - Catcoin

```
INFO:Detectors:
Catcoin.allowance(address,address).owner (CatCoin.sol#230) shadows:
        - Ownable.owner() (CatCoin.sol#50-52) (function)
Catcoin._approve(address,address,uint256).owner (CatCoin.sol#351) shadows:
        - Ownable.owner() (CatCoin.sol#50-52) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Catcoin.constructor(address)._pair (CatCoin.sol#184-185) lacks a zero-check on :
                - pair = _pair (CatCoin.sol#188)
Catcoin.updateRouterAndPair(address,address).newPair (CatCoin.sol#558) lacks a zero-check on :
                - pair = newPair (CatCoin.sol#560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Catcoin._transfer(address,address,uint256) (CatCoin.sol#358-416):
        External calls:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
        State variables written after the call(s):
        - taxFreeTransfer(from,to,amount) (CatCoin.sol#412)
                - _rExcludedFromRewardTotal += tAmount * rate (CatCoin.sol#585)
                - _rExcludedFromRewardTotal -= tAmount * rate (CatCoin.sol#595)
        - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
                - _rExcludedFromRewardTotal += tAmount * rate (CatCoin.sol#585)
                - _rExcludedFromRewardTotal -= tAmount * rate (CatCoin.sol#595)
        - taxFreeTransfer(from,to,amount) (CatCoin.sol#412)
                - _tExcludedFromRewardTotal += tAmount (CatCoin.sol#584)
                - _tExcludedFromRewardTotal -= tAmount (CatCoin.sol#594)
        - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
                - _tExcludedFromRewardTotal += tAmount (CatCoin.sol#584)
```

```
INFO:Detectors:
Catcoin.allowance(address,address).owner (CatCoin.sol#230) shadows:
        - Ownable.owner() (CatCoin.sol#50-52) (function)
Catcoin._approve(address,address,uint256).owner (CatCoin.sol#351) shadows:
        - Ownable.owner() (CatCoin.sol#50-52) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Catcoin.constructor(address)._pair (CatCoin.sol#184-185) lacks a zero-check on :
              - pair = _pair (CatCoin.sol#188)
Catcoin.updateRouterAndPair(address,address).newPair (CatCoin.sol#558) lacks a zero-check on :
              - pair = newPair (CatCoin.sol#560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Catcoin._transfer(address,address,uint256) (CatCoin.sol#358-416):
        External calls:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
              - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
              - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
              - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
        State variables written after the call(s):
        - taxFreeTransfer(from,to,amount) (CatCoin.sol#412)
              - _rExcludedFromRewardTotal += tAmount * rate (CatCoin.sol#585)
              - _rExcludedFromRewardTotal -= tAmount * rate (CatCoin.sol#595)
        - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
              - _rExcludedFromRewardTotal += tAmount * rate (CatCoin.sol#585)
              - _rExcludedFromRewardTotal -= tAmount * rate (CatCoin.sol#595)
        - taxFreeTransfer(from,to,amount) (CatCoin.sol#412)
              - _tExcludedFromRewardTotal += tAmount (CatCoin.sol#584)
              - _tExcludedFromRewardTotal -= tAmount (CatCoin.sol#594)
        - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
              - _tExcludedFromRewardTotal += tAmount (CatCoin.sol#584)
```

```
            - _rOwned[account] = 0 (CatCoin.sol#278)
        - _rOwned[owner()] = _rTotal (CatCoin.sol#193)
        - excludeFromReward(pair) (CatCoin.sol#190)
            - _tExcludedFromRewardTotal += tBalance (CatCoin.sol#279)
        - excludeFromReward(DEAD_ADDRESS) (CatCoin.sol#191)
            - _tExcludedFromRewardTotal += tBalance (CatCoin.sol#279)
        - excludeFromReward(pair) (CatCoin.sol#190)
            - _tOwned[account] = tBalance (CatCoin.sol#277)
        - excludeFromReward(DEAD_ADDRESS) (CatCoin.sol#191)
            - _tOwned[account] = tBalance (CatCoin.sol#277)
        - pair = _pair (CatCoin.sol#188)
        - router = _router (CatCoin.sol#187)
Reentrancy in Catcoin.swapAndLiquify(uint256) (CatCoin.sol#476-484):
        External calls:
        - swapTokensForBNB(tokensToSwap,address(this)) (CatCoin.sol#481)
            - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        - addLiquidity(otherHalfOfTokens,newBalance) (CatCoin.sol#483)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
        External calls sending eth:
        - addLiquidity(otherHalfOfTokens,newBalance) (CatCoin.sol#483)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
        State variables written after the call(s):
        - addLiquidity(otherHalfOfTokens,newBalance) (CatCoin.sol#483)
            - _allowances[owner][spender] = amount (CatCoin.sol#354)
Reentrancy in Catcoin.transferFrom(address,address,uint256) (CatCoin.sol#239-247):
        External calls:
        - _transfer(sender,recipient,amount) (CatCoin.sol#240)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
            - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (CatCoin.sol#240)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
```

```
        - _approve(sender,_msgSender(),currentAllowance - amount) (CatCoin.sol#244)
            - _allowances[owner][spender] = amount (CatCoin.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Catcoin._transfer(address,address,uint256) (CatCoin.sol#358-416):
        External calls:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
            - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(swapTokensAtAmount) (CatCoin.sol#406)
            - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tAmount) (CatCoin.sol#578)
            - taxFreeTransfer(from,to,amount) (CatCoin.sol#412)
        - Transfer(sender,address(this),tLiquidity) (CatCoin.sol#431)
            - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
        - Transfer(sender,marketingAddress,tMarketing) (CatCoin.sol#440)
            - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
        - Transfer(sender,DEAD_ADDRESS,tBurn) (CatCoin.sol#450)
            - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
        - Transfer(sender,recipient,tTransferAmount) (CatCoin.sol#468)
            - _tokenTransfer(from,to,amount,usedTaxes) (CatCoin.sol#414)
Reentrancy in Catcoin.constructor(address) (CatCoin.sol#182-202):
        External calls:
        - _pair = IFactory(_router.factory()).createPair(address(this),_router.WETH()) (CatCoin.sol#184-185)
        Event emitted after the call(s):
        - Transfer(address(0),owner(),T_TOTAL) (CatCoin.sol#201)
Reentrancy in Catcoin.swapAndLiquify(uint256) (CatCoin.sol#476-484):
        External calls:
        - swapTokensForBNB(tokensToSwap,address(this)) (CatCoin.sol#481)
            - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
) (CatCoin.sol#510-516)
        - addLiquidity(otherHalfOfTokens,newBalance) (CatCoin.sol#483)
```

```
                    - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (CatCoin.sol
1-498)
         Event emitted after the call(s):
         - Approval(owner,spender,amount) (CatCoin.sol#355)
                    - _approve(sender,_msgSender(),currentAllowance - amount) (CatCoin.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Catcoin._transfer(address,address,uint256) (CatCoin.sol#358-416) uses timestamp for comparisons
         Dangerous comparisons:
         - newCycle = block.timestamp - userLastSell[from].lastSellTime >= antiDumpCycle (CatCoin.sol#392)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Catcoin.includeInReward(address) (CatCoin.sol#286-315) has costly operations inside a loop:
         - _rTotal += rBalance - _rExcludedFromRewardTotal (CatCoin.sol#297)
Catcoin.includeInReward(address) (CatCoin.sol#286-315) has costly operations inside a loop:
         - _rExcludedFromRewardTotal = 0 (CatCoin.sol#301)
Catcoin.includeInReward(address) (CatCoin.sol#286-315) has costly operations inside a loop:
         - _tExcludedFromRewardTotal -= tBalance (CatCoin.sol#308)
Catcoin.includeInReward(address) (CatCoin.sol#286-315) has costly operations inside a loop:
         - _rTotal -= _rExcludedFromRewardTotal - rBalance (CatCoin.sol#299)
Catcoin.includeInReward(address) (CatCoin.sol#286-315) has costly operations inside a loop:
         - _rExcludedFromRewardTotal -= rBalance (CatCoin.sol#304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
Context._msgData() (CatCoin.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Catcoin._reflectionRate (CatCoin.sol#122) is set pre-construction with a non-constant function or state variable:
         - _rTotal / T_TOTAL
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.4 (CatCoin.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Variable Catcoin._rExcludedFromRewardTotal (CatCoin.sol#125) is too similar to Catcoin._tExcludedFromRewardTotal (CatCoin.sol#
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Catcoin.slitherConstructorConstantVariables() (CatCoin.sol#99-664) uses literals with too many digits:
         - DEAD_ADDRESS = 0x000000000000000000000000000000000000dEaD (CatCoin.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
         - Ownable.renounceOwnership() (CatCoin.sol#59-61)
transferOwnership(address) should be declared external:
         - Ownable.transferOwnership(address) (CatCoin.sol#63-66)
name() should be declared external:
         - Catcoin.name() (CatCoin.sol#205-207)
symbol() should be declared external:
         - Catcoin.symbol() (CatCoin.sol#208-210)
decimals() should be declared external:
         - Catcoin.decimals() (CatCoin.sol#211-213)
totalSupply() should be declared external:
         - Catcoin.totalSupply() (CatCoin.sol#216-218)
transfer(address,uint256) should be declared external:
         - Catcoin.transfer(address,uint256) (CatCoin.sol#225-228)
allowance(address,address) should be declared external:
         - Catcoin.allowance(address,address) (CatCoin.sol#230-232)
approve(address,uint256) should be declared external:
         - Catcoin.approve(address,uint256) (CatCoin.sol#234-237)
transferFrom(address,address,uint256) should be declared external:
         - Catcoin.transferFrom(address,address,uint256) (CatCoin.sol#239-247)
increaseAllowance(address,uint256) should be declared external:
         - Catcoin.increaseAllowance(address,uint256) (CatCoin.sol#249-252)
decreaseAllowance(address,uint256) should be declared external:
         - Catcoin.decreaseAllowance(address,uint256) (CatCoin.sol#254-260)
isExcludedFromReward(address) should be declared external:
         - Catcoin.isExcludedFromReward(address) (CatCoin.sol#262-264)
```

```
excludeFromFee(address) should be declared external:
        - Catcoin.excludeFromFee(address) (CatCoin.sol#318-320)
includeInFee(address) should be declared external:
        - Catcoin.includeInFee(address) (CatCoin.sol#322-324)
isExcludedFromFee(address) should be declared external:
        - Catcoin.isExcludedFromFee(address) (CatCoin.sol#327-329)
isBot(address) should be declared external:
        - Catcoin.isBot(address) (CatCoin.sol#569-571)
rescueAnyBEP20Tokens(address,address,uint256) should be declared external:
        - Catcoin.rescueAnyBEP20Tokens(address,address,uint256) (CatCoin.sol#656-659)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CatCoin.sol analyzed (6 contracts with 75 detectors), 56 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis - Catcoin

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Catcoin.swapTokensForBNB(uint256,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 501:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 515:12:

Gas & Economy

Gas costs:

Gas requirement of function Catcoin.transferTaxes is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 164:4:

Gas costs:

Gas requirement of function Catcoin.rescueAnyBEP20Tokens is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 656:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 603:8:

Miscellaneous

Constant/View/Pure functions:

IRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[
: Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 91:4:

Similar variable names:

Catcoin.airdropTokens(address[],uint256[]) : Variables have very similar names "accounts" and "amounts". Note: Modifiers are currently not considered by this static analysis.
Pos: 604:53:

Similar variable names:

Catcoin.airdropTokens(address[],uint256[]) : Variables have very similar names "accounts" and "amounts". Note: Modifiers are currently not considered by this static analysis.
Pos: 603:30:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 657:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 478:31:

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Update: The ownership of the contract is renounced to 0x000.....dEaD address using transferOwnership function, hence this deployed contract is now fully decentralized without any ownership control. All ownerOnly functions have been removed..

Tx hash:

https://bscscan.com/tx/0x882cebbea41232773cd0b7b85dfb5ea02e9ceab59fe7cf5ec6e0e78bc64a83be

Critical:

No critical severity vulnerabilities were found.

High:

No medium severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Discussion

1) Instead of using a large number of mappings, a single struct can reduce the gas amount.

2) Unused function declaration inside IERC20 interface can be removed.

3) Function _msgData under context is nowhere used.

4) Transferownership and acceptownership is good practice in case by mistake ownership is transferred to the wrong address.

Conclusion

We were given a contract file and have used all possible tests based on the given object. The contract is written systematically, it is ready to go for production.

We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is now "secured".

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

RD Auditors Disclaimer

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

RD
AUDITORS

Email: info@rdauditors.com

Website: www.rdauditors.com