Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Catch Coin
Website:	www.catchcoin.com
Platform:	Base Chain
Language:	Solidity
Date:	May 3rd, 2024

Table of contents

Email: audit@EtherAuthority.io

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Catch Coin team to perform the Security audit of the Catch Coin smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 3rd, 2024.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- The Solidity contract for "CATCHCOIN" implements the ERC20 standard.Here's a breakdown of its key features and functionalities:
    - **ERC20 Implementation:** The contract implements the ERC20 interface with standard functions such as totalSupply, balanceOf, transfer, allowance, approve, and transferFrom.
    - **Ownable:** The contract includes an Ownable implementation, allowing the owner to perform certain privileged actions such as transferring ownership and renouncing ownership.
    - **Uniswap Integration:** The contract integrates with Uniswap V2 by defining interfaces for the Uniswap V2 Factory and Router contracts. It provides functions for creating a pair, adding liquidity, and swapping tokens.
    - **Fee Mechanism:** The contract includes a fee mechanism where fees are charged on transfers. There are different types of fees such as reflection fee, liquidity fee, coin operation fee, and burn fee. The fees are calculated based on the transferred amount.
    - **Exclusion Mechanism:** The contract allows certain addresses to be excluded from fees or from reflection rewards.
    - **Trading Restrictions:** There are conditions set for trading, such as enabling trading only after a certain time and enabling trading only for certain addresses.

- **Automatic Liquidity:** The contract automatically adds liquidity to the Uniswap pool on each transaction if certain conditions are met.
- **Reflection:** The contract implements a reflection mechanism where holders receive rewards based on the amount of tokens they hold.
- **Airdrop Functionality:** There's a function to perform airdrops to multiple addresses.
- **Modifiers and Events:** Modifiers like onlyOwner and lockTheSwap are used to restrict access to certain functions. Events are emitted for important contract actions.

- Overall, this contract facilitates token transfers, fees collection, liquidity provision, and rewards distribution while integrating with the Uniswap decentralized exchange for liquidity management.

Audit scope

Name	**Code Review and Security Analysis Report for Catch Coin Smart Contract**
Platform	**Base Chain**
Language	**Solidity**
File	CATCHCOIN.sol
Smart Contract Code	0x95017e6f16375e63e5cb4d3a5fbf3c40775b08f4
Audit Date	May 3rd, 2024

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: catchcoin • Symbol: CATCH • Decimals: 18	**YES, This is valid.**
Controller of the owner control: • Current owner can transfer the ownership. • Owner can renounce ownership. • Grants the owner the ability to exclude an address from transaction fees. • Grants the owner the ability to include an address in transaction fees. • Set the address of the fund wallet. • Allows the owner to enable or disable the swap and liquify feature. • External function for updating the threshold amount required for triggering liquidity addition. • Start trading. • Airdrop tokens.	**YES, This is valid.** **We advise to renounce ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 2 low and 1 very low level issues.

We confirm that 2 low and 1 very low-severity issues are acknowledged in the smart contract code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Email: audit@EtherAuthority.io

Business Risk Analysis

Category	Result
Buy Tax	0%
Sell Tax	0%
Cannot Buy	No
Cannot Sell	No
Max Tax	0%
Modify Tax	Not Detected
Fee Check	No
Is Honeypot	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	No
Pause Transfer?	No
Max Tax?	No
Is it Anti-whale?	Not Detected
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	No
Can Mint?	No
Is it Proxy?	Not Detected
Can Take Ownership?	Yes
Hidden Owner?	Not Detected
Self Destruction?	Not Detected
Auditor Confidence	High

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Catch Coin are part of its 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 a specific address and its properties / methods can be reused many times by other contracts in the Catch Coin.

The Catch Coin team has not provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a Catch Coin smart contract code in the form of a basescan.org web link.

As mentioned above, code parts are well commented on. And the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	lockTheSwap	modifier	Passed	No Issue
3	name	external	Passed	No Issue
4	symbol	external	Passed	No Issue
5	decimals	external	Passed	No Issue
6	totalSupply	external	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	external	Passed	No Issue
9	allowance	external	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	external	Passed	No Issue
12	increaseAllowance	external	Passed	No Issue
13	decreaseAllowance	external	Passed	No Issue
14	isExcludedFromReward	external	Passed	No Issue
15	totalFees	external	Passed	No Issue
16	deliver	external	Passed	No Issue
17	reflectionFromToken	external	Passed	No Issue
18	tokenFromReflection	read	Passed	No Issue
19	excludeFromReward	external	access only Owner	No Issue
20	includeInReward	external	Infinite loops possibility	Refer Audit Findings
21	_transferBothExcluded	write	Passed	No Issue
22	excludeFromFee	external	access only Owner	No Issue
23	includeInFee	external	access only Owner	No Issue
24	setFundWallet	external	access only Owner	No Issue
25	setSwapAndLiquifyEnabled	external	access only Owner	No Issue
26	updateThreshold	external	access only Owner	No Issue
27	receive	external	Passed	No Issue
28	_reflectFee	write	Passed	No Issue
29	_takeCoinFund	write	Passed	No Issue
30	_getValues	read	Passed	No Issue
31	_getValue	read	Passed	No Issue
32	_getTValues	read	Passed	No Issue
33	_getRValues	read	Passed	No Issue
34	_getRate	read	Passed	No Issue
35	_getCurrentSupply	read	Infinite loops possibility	Refer Audit Findings
36	_takeLiquidity	write	Passed	No Issue
37	calculateTaxFee	read	Passed	No Issue
38	calculateLiquidityFee	read	Passed	No Issue
39	calculateCoinOperartionTax	read	Passed	No Issue
40	calculateBurnTax	read	Passed	No Issue

41	removeAllFee	write	Passed	No Issue
42	isExcludedFromFee	external	Passed	No Issue
43	_approve	write	Passed	No Issue
44	startTrading	external	access only Owner	No Issue
45	_transfer	write	Passed	No Issue
46	airdrop	external	High gas consuming loop in airdrop function	Refer Audit Findings
47	_sellBuyTax	write	Passed	No Issue
48	swapAndLiquify	write	lockTheSwap	No Issue
49	swapTokensForEth	write	Passed	No Issue
50	addLiquidity	write	Passed	No Issue
51	_tokenTransfer	write	Passed	No Issue
52	_transferStandard	write	Passed	No Issue
53	_transferToExcluded	write	Passed	No Issue
54	_transferFromExcluded	write	Passed	No Issue
55	owner	read	Passed	No Issue
56	onlyOwner	modifier	Passed	No Issue
57	renounceOwnership	write	access only Owner	No Issue
58	transferOwnership	write	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

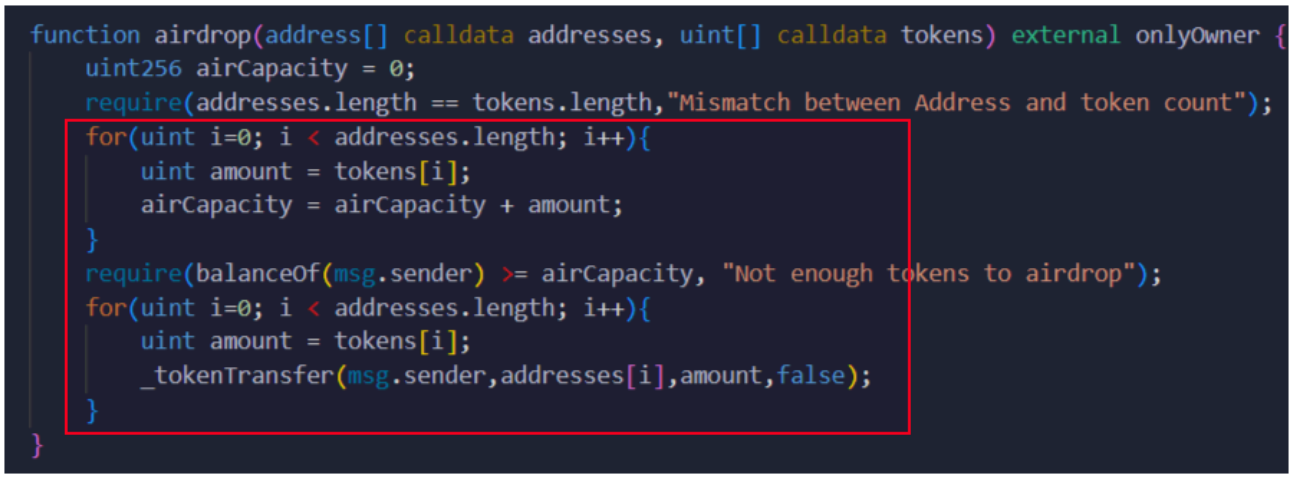
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) High gas consuming loop in airdrop function:



```
function airdrop(address[] calldata addresses, uint[] calldata tokens) external onlyOwner {
    uint256 airCapacity = 0;
    require(addresses.length == tokens.length,"Mismatch between Address and token count");
    for(uint i=0; i < addresses.length; i++){
        uint amount = tokens[i];
        airCapacity = airCapacity + amount;
    }
    require(balanceOf(msg.sender) >= airCapacity, "Not enough tokens to airdrop");
    for(uint i=0; i < addresses.length; i++){
        uint amount = tokens[i];
        _tokenTransfer(msg.sender,addresses[i],amount,false);
    }
}
```

The airdrop function allows the owner to input unlimited wallets. So, the owner must input limited wallets, as inputting excessive wallets might hit the block's gas limit. The owner can accept this risk and can execute this function using limited wallets only.

Resolution: We suggest specifying some limit on the number of wallets can be used. This will prevent any potential human error.
Status: **This issue is acknowledged.**

(2) Infinite loops possibility:

As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.

Resolution: Adjust logic to replace loops with mapping or other code structure.

- includeInReward() - _excluded.length.
- _getCurrentSupply() - _excluded.length.

Status: **This issue is acknowledged.**

Very Low / Informational / Best practices:

(1) Variable can be immutable:

The startingHr variables should be declared immutable to save gas.

Resolution: We suggest marking this variable as immutable.

Status: **This issue is acknowledged.**

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

CATCHCOIN.sol

- excludeFromReward: This function excludes the specified address from receiving reflections by the owner.
- includeInReward: This function for including an account in the reward distribution by the owner.
- excludeFromFee: Grants the owner the ability to exclude an address from transaction fees by the owner.
- includeInFee: Grants the owner the ability to include an address in transaction fees by the owner.
- setFundWallet: The owner can set the address of the fund wallet.
- setSwapAndLiquifyEnabled: Allows the owner to enable or disable the swap and liquify feature by the owner.
- updateThreshold: External function for updating the threshold amount required for triggering liquidity addition by the owner.
- startTrading: The owner can start trading.
- airdrop: The owner can airdrop tokens.

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a basescan.org web link. And we have used all possible tests based on given objects as files. We had observed 2 low and 1 informational issues in the smart contracts. We confirm that 2 low and 1 very low-severity issues are acknowledged in the smart contract code. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

Email: audit@EtherAuthority.io

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 analyze 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 scrutinized 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

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract 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).

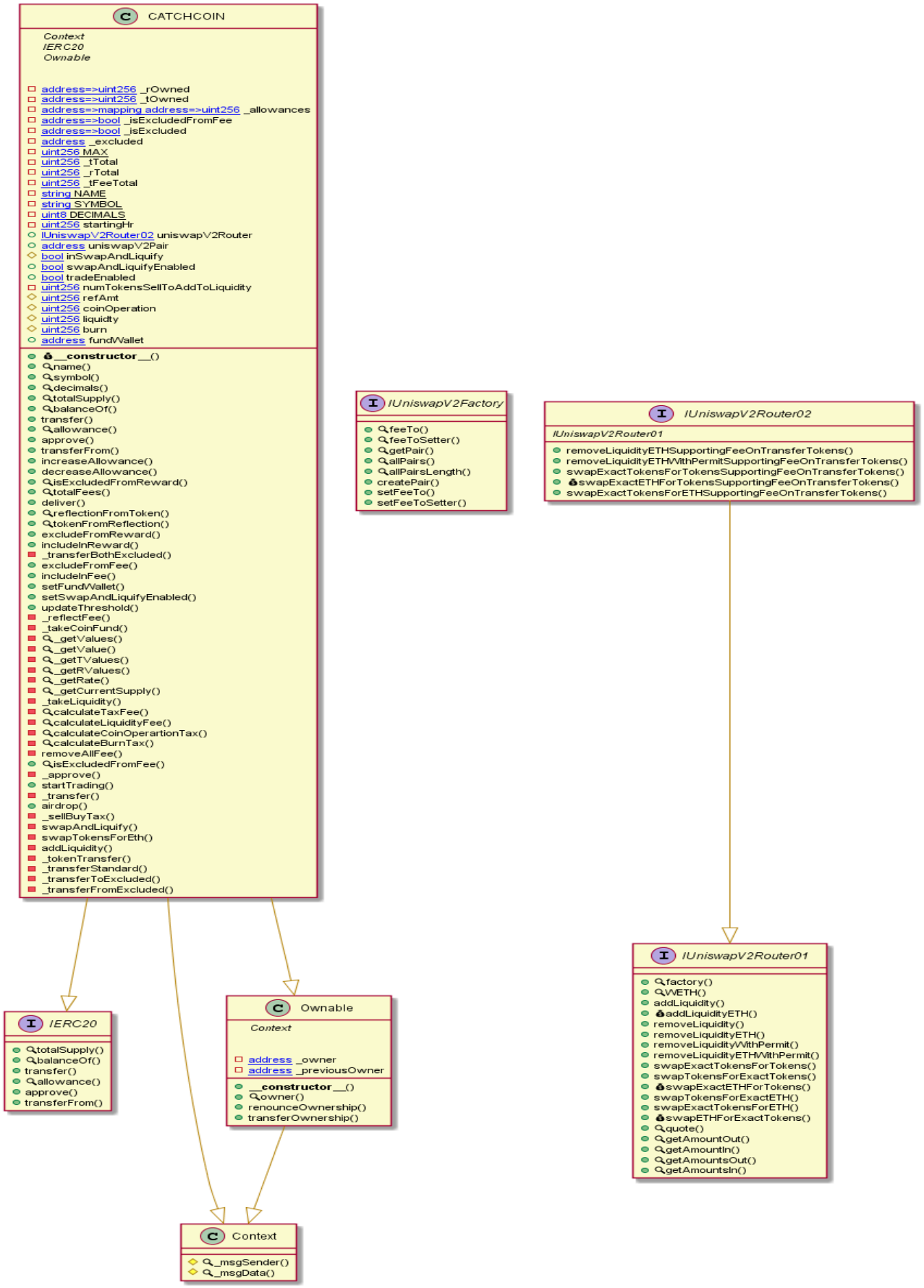
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. 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.

Appendix

Code Flow Diagram - Catch Coin



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither Log >> CATCHCOIN.sol

```
INFO:Detectors:
CATCHCOIN.allowance(address,address).owner (CATCHCOIN.sol#520) shadows:
        - Ownable.owner() (CATCHCOIN.sol#161-163) (function)
CATCHCOIN._approve(address,address,uint256).owner (CATCHCOIN.sol#1055) shadows:
        - Ownable.owner() (CATCHCOIN.sol#161-163) (function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-sha
dowing
INFO:Detectors:
Reentrancy in CATCHCOIN._transfer(address,address,uint256)
(CATCHCOIN.sol#1095-1153):
        External calls:
        - swapAndLiquify(contractTokenBalance) (CATCHCOIN.sol#1118)
                - uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CATCHCOIN.sol#1294-1301)
                -
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()] -
amount) (CATCHCOIN.sol#549)
                - _allowances[owner][spender] = amount (CATCHCOIN.sol#1059)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnera
bilities-2
INFO:Detectors:
Reentrancy in CATCHCOIN._transfer(address,address,uint256)
(CATCHCOIN.sol#1095-1153):
        External calls:
        - swapAndLiquify(contractTokenBalance) (CATCHCOIN.sol#1118)
                - uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CATCHCOIN.sol#1294-1301)

        Event emitted after the call(s):
        - Approval(owner,spender,amount) (CATCHCOIN.sol#1060)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()]
- amount) (CATCHCOIN.sol#549)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnera
bilities-3
INFO:Detectors:
CATCHCOIN._transfer(address,address,uint256) (CATCHCOIN.sol#1095-1153) uses
timestamp for comparisons
```

```
        Dangerous comparisons:
        - startingHr >= block.timestamp (CATCHCOIN.sol#1133)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
CATCHCOIN.includeInReward(address) (CATCHCOIN.sol#678-689) has costly operations
inside a loop:
        - _excluded.pop() (CATCHCOIN.sol#685)
CATCHCOIN.removeAllFee() (CATCHCOIN.sol#1030-1035) has costly operations inside
a loop:
        - refAmt = 0 (CATCHCOIN.sol#1031)
CATCHCOIN.removeAllFee() (CATCHCOIN.sol#1030-1035) has costly operations inside
a loop:
        - coinOperation = 0 (CATCHCOIN.sol#1032)
CATCHCOIN.removeAllFee() (CATCHCOIN.sol#1030-1035) has costly operations inside
a loop:
        - liquidty = 0 (CATCHCOIN.sol#1033)
CATCHCOIN.removeAllFee() (CATCHCOIN.sol#1030-1035) has costly operations inside
a loop:
        - burn = 0 (CATCHCOIN.sol#1034)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CATCHCOIN.sol#805-815) has
costly operations inside a loop:
        - _rTotal = _rTotal - rFee - rBurn (CATCHCOIN.sol#809)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CATCHCOIN.sol#805-815) has
costly operations inside a loop:
        - _tFeeTotal = _tFeeTotal + tFee (CATCHCOIN.sol#810)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CATCHCOIN.sol#805-815) has
costly operations inside a loop:
        - _tTotal = _tTotal - tBurn (CATCHCOIN.sol#812)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-
inside-a-loop
INFO:Detectors:
Context._msgData() (CATCHCOIN.sol#124-127) is never used and should be removed
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
CATCHCOIN._rTotal (CATCHCOIN.sol#371) 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-initializ
ing-state
INFO:Detectors:
Pragma version0.8.19 (CATCHCOIN.sol#42) 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:
Function IUniswapV2Router01.WETH() (CATCHCOIN.sol#225) is not in mixedCase
Parameter CATCHCOIN.setFundWallet(address)._fundWallet (CATCHCOIN.sol#753) is
not in mixedCase
Parameter CATCHCOIN.setSwapAndLiquifyEnabled(bool)._enabled (CATCHCOIN.sol#765)
is not in mixedCase
Parameter CATCHCOIN.updateThreshold(uint256)._amount (CATCHCOIN.sol#781) is not
in mixedCase
Parameter CATCHCOIN.calculateTaxFee(uint256)._amount (CATCHCOIN.sol#981) is not
in mixedCase
Parameter CATCHCOIN.calculateLiquidityFee(uint256)._amount (CATCHCOIN.sol#992)
is not in mixedCase
Parameter CATCHCOIN.calculateCoinOperartionTax(uint256)._amount
(CATCHCOIN.sol#1004) is not in mixedCase
Parameter CATCHCOIN.calculateBurnTax(uint256)._amount (CATCHCOIN.sol#1017) is
```

```
not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-sol
idity-naming-conventions
INFO:Detectors:
Redundant expression "this (CATCHCOIN.sol#125)" inContext
(CATCHCOIN.sol#119-128)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statemen
ts
INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,
address,uint256).amountADesired (CATCHCOIN.sol#230) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,
address,uint256).amountBDesired (CATCHCOIN.sol#231)
Variable
CATCHCOIN._getRValues(uint256,uint256,uint256,uint256,uint256).rCoinOperation
(CATCHCOIN.sol#920) is too similar to
CATCHCOIN._transferToExcluded(address,address,uint256).tCoinOperation
(CATCHCOIN.sol#1373)
Variable
CATCHCOIN._getRValues(uint256,uint256,uint256,uint256,uint256).rCoinOperation
(CATCHCOIN.sol#920) is too similar to
CATCHCOIN._getTValues(uint256).tCoinOperation (CATCHCOIN.sol#893)
Variable
CATCHCOIN._transferBothExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#709) is too similar to
CATCHCOIN._transferStandard(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1350)
Variable CATCHCOIN.reflectionFromToken(uint256,bool).rTransferAmount
(CATCHCOIN.sol#631) is too similar to
CATCHCOIN._transferToExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1373)
Variable
CATCHCOIN._transferFromExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1399) is too similar to
CATCHCOIN._getValues(uint256).tTransferAmount (CATCHCOIN.sol#854)
Variable
CATCHCOIN._transferBothExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#709) is too similar to
CATCHCOIN._transferToExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1373)
Variable CATCHCOIN._getValue(uint256).rTransferAmount (CATCHCOIN.sol#871) is too
similar to
CATCHCOIN._transferBothExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#708)
Variable
CATCHCOIN._transferFromExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1399) is too similar to
CATCHCOIN._transferToExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1373)
Variable CATCHCOIN._transferStandard(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1351) is too similar to
CATCHCOIN._transferToExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1373)
Variable CATCHCOIN.reflectionFromToken(uint256,bool).rTransferAmount
(CATCHCOIN.sol#631) is too similar to
CATCHCOIN._transferBothExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#708)
Variable
CATCHCOIN._transferBothExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#709) is too similar to
CATCHCOIN._transferBothExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#708)
```

```
Variable
CATCHCOIN._transferFromExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1399) is too similar to
CATCHCOIN._transferBothExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#708)
Variable CATCHCOIN._getValue(uint256).rTransferAmount (CATCHCOIN.sol#871) is too
similar to CATCHCOIN._getValues(uint256).tTransferAmount (CATCHCOIN.sol#854)
Variable CATCHCOIN._transferStandard(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1351) is too similar to
CATCHCOIN._transferBothExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#708)
Variable CATCHCOIN.reflectionFromToken(uint256,bool).rTransferAmount
(CATCHCOIN.sol#631) is too similar to
CATCHCOIN._transferFromExcluded(address,address,uint256).rTransferAmount
(CATCHCOIN.sol#1399) is too similar to
CATCHCOIN._transferFromExcluded(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1398)
Variable CATCHCOIN._getValue(uint256).rTransferAmount (CATCHCOIN.sol#871) is too
similar to CATCHCOIN._transferStandard(address,address,uint256).tTransferAmount
(CATCHCOIN.sol#1350)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too
-similar
INFO:Detectors:
Loop condition i < _excluded.length (CATCHCOIN.sol#949) should use cached array
length instead of referencing a `length` member of the storage array.
 Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
CATCHCOIN.startingHr (CATCHCOIN.sol#379) should be immutable
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-th
at-could-be-declared-immutable
INFO:Slither:CATCHCOIN.sol analyzed (7 contracts with 93 detectors), 95
result(s) found
```

Solidity Static Analysis

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

CATCHCOIN.sol

Check-effects-interaction:
Potential violation of Checks-Effects-Interaction pattern in CATCHCOIN.(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
Pos: 437:47:

Check-effects-interaction:
Potential violation of Checks-Effects-Interaction pattern in CATCHCOIN.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
Pos: 1278:2:

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.
Pos: 1314:28:

Gas costs:
Gas requirement of function CATCHCOIN.includeInReward 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: 694:29:

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 a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
Pos: 965:3:

ERC20:
ERC20 contract's "decimals" function should have "uint8" as return type
Pos: 490:67:

Constant/View/Pure functions:
CATCHCOIN.reflectionFromToken(uint256,bool) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
Pos: 639:64:

Similar variable names:
CATCHCOIN.(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 441:18:

No return:
IUniswapV2Router02.removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32): Defines a return type but never explicitly returns a value.
Pos: 347:13:

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.
Pos: 744:3:

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: 952:14:

Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

CATCHCOIN.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:41
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:151
Error message for require is too long
Pos: 9:189
Function name must be in mixedCase
Pos: 5:224
Contract has 19 states declarations but allowed no more than 15
Pos: 1:358
Explicitly mark visibility of state
Pos: 5:392
Explicitly mark visibility of state
Pos: 5:393
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 8:423
Avoid making time-based decisions in your business logic
Pos: 22:441
Error message for require is too long
Pos: 9:781
Code contains empty blocks
Pos: 9:1097
Provide an error message for require
Pos: 9:1104
Avoid making time-based decisions in your business logic
Pos: 30:1132
Error message for require is too long
Pos: 9:1156
Avoid making time-based decisions in your business logic
Pos: 13:1271
Avoid making time-based decisions in your business logic
Pos: 13:1299
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io