EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	catchcoin
Website:	catchcoin.com
Platform:	Base Chain Network
Language:	Solidity
Date:	February 14th, 2025

Table of contents

Introduction

EtherAuthority was contracted by catchcoin to perform the Security audit of the CATCH Token smart contract code. The audit was performed using manual analysis and automated software tools. This report presents all the findings regarding the audit performed on February 14th, 2025.

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

CATCHCoin is a reflection-based ERC20 token with automated tax distribution, liquidity management, and trading control mechanisms. It integrates with Uniswap V2 for automated market-making and includes features such as:

- **Reflection Mechanism**: Rewards token holders automatically through transaction fees.
- **Liquidity Management**: Automatically adds liquidity via Uniswap V2.
- **Tax and Fee Structure**: Supports buy/sell tax, reflection fees, liquidity fees, coin operation tax, and burn tax.
- **Trading Controls**: These include trading enable/disable functionality and an initial restriction period for anti-bot protection.
- **Exclusion Mechanism**: Allows specific addresses to be excluded from fees and rewards.
- **Fund Wallet**: A dedicated wallet for operational costs, funded by transaction fees.
- **Swap and Liquify**: Converts collected fees into liquidity to maintain a stable market.
- **Ownership and Access Control**: Only the owner can modify tax rates, exclude addresses, and enable trading.

CATCHCoin is designed for sustainability and fair token distribution, ensuring a balance between rewards, liquidity, and ecosystem growth.

Audit scope

Name	**Code Review and Security Analysis Report for catchcoin Smart Contract**
Platform	**Base Chain Network**
Language	**Solidity**
File	CatchCoin.sol
Initial Code Link	0x44f77502418c9b225ad8f80a9def915c8c271a19
Audit Date	February 14th, 2025

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Token Details: • Name: catchcoin • Symbol: CATCH • Decimals: 18 • Total Supply: 180 Million CATCH tokens.	**YES, This is valid.**
Key Components: **1. Dual Accounting & Reflection:** • **Reflection vs. Token Balances:** ○ The contract uses two parallel balances: ■ **_rOwned:** Reflection balances for every address. ■ **_tOwned:** Actual token balances for addresses excluded from the reward mechanism. • **Reward Distribution:** ○ A portion of every transaction is taken as a fee (the reflection fee) and redistributed to all holders via an adjustment of the total reflections (_rTotal). This process is managed by helper functions like _reflectFee, which reduce the reflection supply and update the total fees collected. • **Conversion Helpers:** ○ Functions like tokenFromReflection and _getRate allow conversion between the reflection (R-values) and actual token (t-values) balances.	**YES, This is valid. We suggest renouncing ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

2. Fee Structure:

- The contract imposes several fees on transfers, which are calculated as percentages of the transaction amount:
- **Reflection Fee (refAmt):**
    - A fee that is redistributed to token holders.
- **Liquidity Fee (liquidty):**
    - A fee accumulated within the contract is later used to add liquidity to a Uniswap pool.
- **Coin Operation Fee (coinOperation):**
    - A fee that is sent to a designated fund wallet, potentially used for development, marketing, or other operational costs.
- **Burn Fee (burn):**
    - A fee that permanently removes tokens from the total supply, creating a deflationary effect.

<u>**Dynamic Fee Application:**</u>

- **Buy vs. Sell Fees:**
    - The function _sellBuyTax sets different fee rates based on the transaction type.
    - **Buy Transactions (when tokens are purchased from the liquidity pool):**
        - Reflection Fee: 1%
        - Coin Operation Fee: 1%
        - Liquidity Fee: 1%
        - Burn Fee: 0%
    - **Sell Transactions (when tokens are sold back to the liquidity pool):**
        - Reflection Fee: 2%

        - Coin Operation Fee: 1%
        - Liquidity Fee: 2%
        - Burn Fee: 1%
- **Early Trading Period:**
    - The contract sets a startingHr variable (current time plus 4 hours) during deployment. For transactions that occur before this period expires, additional or altered fee logic is applied:
        - For example, on buys during this period, half of the transaction amount is taken as an extra tax and transferred to the contract address. This measure is likely intended to deter bots and early speculative trading.

3. Automatic Liquidity (Swap and Liquify):

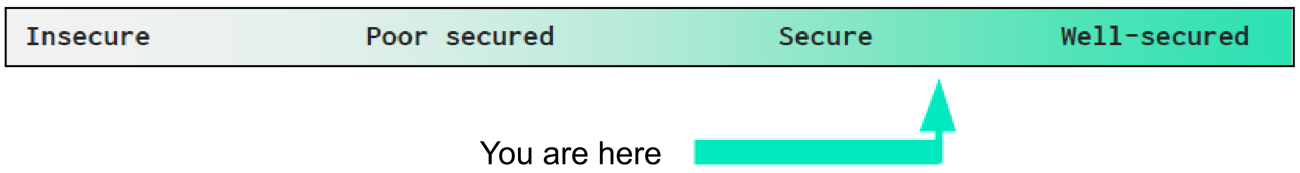
- **Liquidity Threshold:**
    - The contract accumulates tokens from liquidity fees. Once the token balance held by the contract reaches a certain threshold (numTokensSellToAddToLiquidity), the swap and liquify process is triggered.
- **Swap and Liquify Process:**
    - The function swapAndLiquify performs the following steps:
        - **Splitting Tokens:** The accumulated tokens are split into two halves.
        - **Swapping for ETH:** One-half is swapped for ETH using the Uniswap router.

■ **Adding Liquidity:** The other half of the tokens is paired with the acquired ETH and added to the liquidity pool. ○ This process is guarded by the lockTheSwap modifier to prevent reentrancy issues. **4. Trading Control and Anti-Bot Measures:** ● **Trading Enablement:** ○ Trading is controlled by a boolean flag tradeEnabled. Until the owner explicitly calls startTrading, no transfers (apart from those by the owner) can take place. ● **Initial Trading Restrictions:** ○ During the first 4 hours after deployment, the contract applies special fee logic to discourage bots and rapid trading. This is enforced by comparing the current block timestamp with the startingHr value. **5. Exclusions from Fees and Rewards:** ● **Fee Exclusion:** ○ Certain addresses (like the owner or the contract itself) can be excluded from paying fees. This is managed using the _isExcludedFromFee mapping, with functions excludeFromFee and includeInFee to update the status. ● **Reward Exclusion:** ○ Addresses can also be excluded from receiving reflections. The _isExcluded mapping and functions excludeFromReward/includeInReward	

manage this feature. For excluded addresses, the contract maintains an actual token balance (_tOwned) rather than relying on the reflection mechanism. **6. Airdrop Functionality:** ● **Bulk Transfers:** ○ The airdrop function allows the owner to send tokens to multiple addresses in one transaction. It first verifies that the total airdrop amount does not exceed the sender's balance and then performs individual transfers using the _tokenTransfer function. **7. Uniswap Integration** ● **Router and Pair Initialization:** ○ During contract deployment, the contract: ■ Sets the Uniswap V2 router (using a provided address on the Base network). ■ Creates a liquidity pair for the token and the network's wrapped ETH (WETH) via the Uniswap factory. ● **Liquidity Operations:** ○ The functions swapTokensForEth and addLiquidity facilitate token swaps and liquidity additions on Uniswap, using the standard Uniswap V2 router interface.	

Audit Summary

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **"Technically Secured"**. Also, this contract contains owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint, and Remix IDE. At the same time, this finding is based on a 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. The general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

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

Investor Advice: A 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	The solidity version is not specified	Passed
	The solidity version is too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack 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	Passed
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	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage is not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Business Risk Analysis

Category	Result
● Buy Tax	3% (Reflection: 1%, Liquidity: 1%, Coin Operation: 1%)
● Sell Tax	6% (Reflection: 2%, Liquidity: 2%, Coin Operation: 1%, Burn: 1%)
● Cannot Buy	No
● Cannot Sell	No
● Max Tax	No
● Modify Tax	Yes (adjustable by owner)
● Fee Check	Yes (Tax fees are verified and processed per transaction)
● Is Honeypot	Not Detected
● Trading Cooldown	Not Detected
● Can Pause Trade?	Yes (adjustable by owner)
● Pause Transfer?	Not Detected
● Max Transaction amount?	No
● Is it Anti-whale?	No
● Is Anti-bot?	Not Detected
● Is it a Blacklist?	Yes (Owner can blacklist addresses)
● Blacklist Check	Yes
● Can Mint?	No
● Is it a Proxy?	No
● 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 contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in the CATCH Token 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 Token.

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

The smart contracts comment on code parts well commented on, using Ethereum's NatSpec commenting style, which is good.

Documentation

We were given a CATCH Token smart contract code in the form of a basescan.org weblink.

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

Use of Dependencies

As per our observation, the libraries used in this smart contract infrastructure 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	access only Owner	No Issue
21	_transferBothExcluded	write	Passed	No Issue
22	excludeFromFee	external	access only Owner	No Issue
23	includeInFee	external	access only Owner	No Issue
24	setFundWallet1	external	access only Owner	No Issue
25	setFundWallet2	external	access only Owner	No Issue
26	setSwapAndLiquifyEnabled	external	access only Owner	No Issue
27	updateThreshold	external	access only Owner	No Issue
28	receive	external	Passed	No Issue
29	_reflectFee	write	Passed	No Issue
30	_takeCoinFund	write	Passed	No Issue
31	_getValues	read	Passed	No Issue
32	_getValue	read	Passed	No Issue
33	_getTValues	read	Passed	No Issue
34	_getRValues	read	Passed	No Issue
35	_getRate	read	Passed	No Issue
36	_getCurrentSupply	read	Passed	No Issue
37	_takeLiquidity	write	Passed	No Issue
38	calculateTaxFee	read	Passed	No Issue
39	calculateLiquidityFee	read	Passed	No Issue
40	calculateCoinOperartionTax	read	Passed	No Issue
41	calculateBurnTax	read	Passed	No Issue
42	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	access only Owner	No Issue
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 a 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 a 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.

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

No Low-severity vulnerabilities were found.

Very Low / Informational / Best practices:

No informational severity vulnerabilities were found.

Centralization

This smart contract has some functions that can be executed by the Admin (Owner) only. If the admin wallet's private key is compromised, then it creates trouble. The following are Admin functions:

CatchCoin.sol

- excludeFromReward: Allows the owner the ability to exclude an address from earning reflections.
- includeInReward: Grants the owner the ability to include an account in the reward distribution.
- excludeFromFee: Grants the owner the ability to exclude an address from transaction fees.
- includeInFee: Grants the owner the ability to include an address in transaction fees.
- setFundWallet1: Allows the owner to set the address of the fundwallet1.
- setFundWallet2: Allows the owner to set the address of the fundwallet2.
- setSwapAndLiquifyEnabled: Allows the owner to enable or disable the swap and liquify feature.
- updateThreshold: Allows the owner to update the threshold amount required for triggering liquidity addition.
- startTrading: Enables trading for the token by the owner.
- airdrop: Airdrops tokens to multiple addresses by the owner.

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

Conclusion

We were given a contract code as a basescan.org weblink and used all possible tests based on the given objects. We have not observed any issues. **So, the smart contract is ready for mainnet deployment.**

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 the 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.

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

The security state of the reviewed smart contract, based on standard audit procedure scope, is **"Technically 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 the 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 white box penetration testing. We look at the project's website to get a high-level understanding of the functionality of the software under review. 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, and 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).

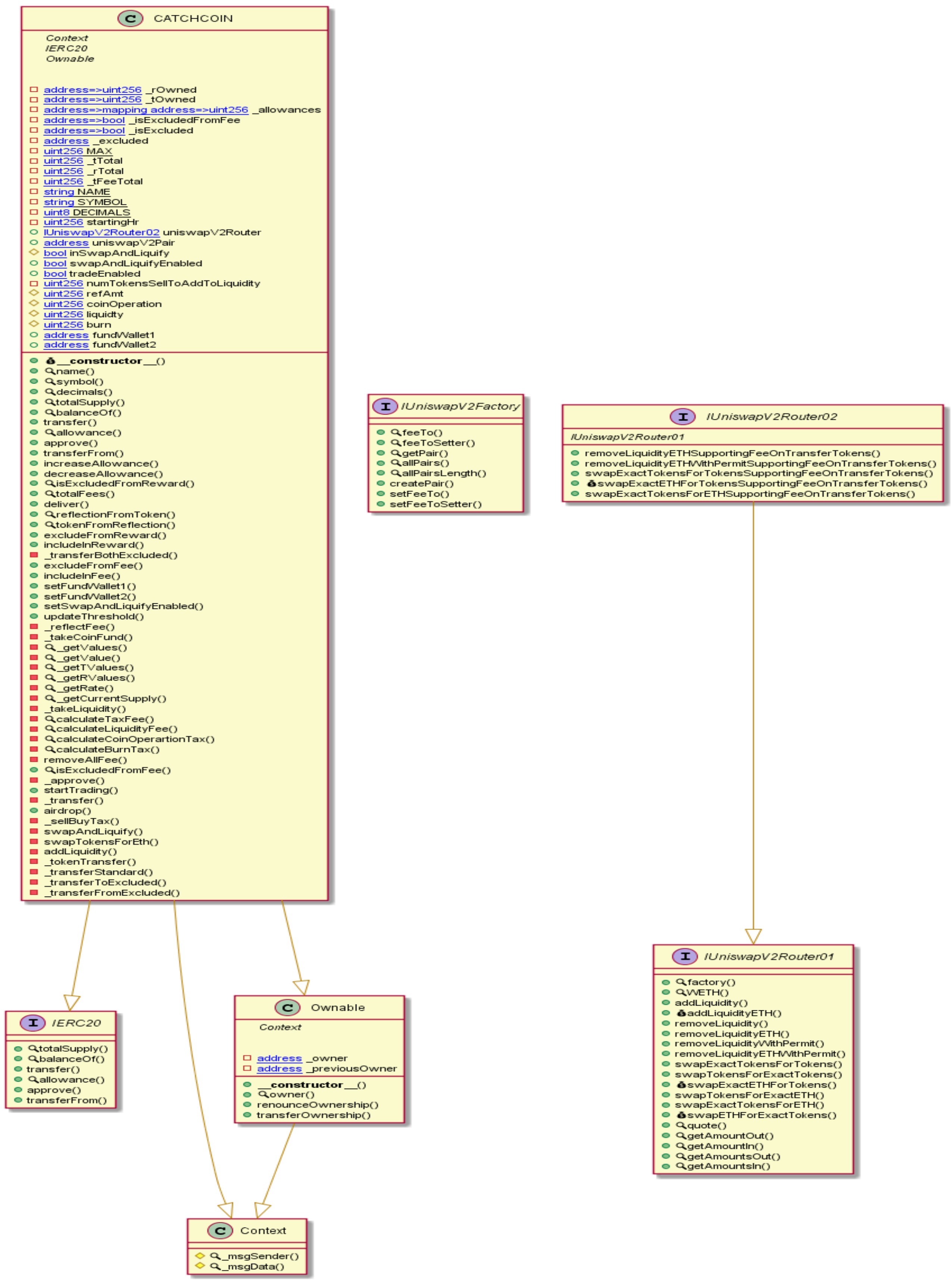
Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best to conduct the analysis and produce 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 the explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - catchcoin



Slither Results Log

Slither Log >> CatchCoin.sol

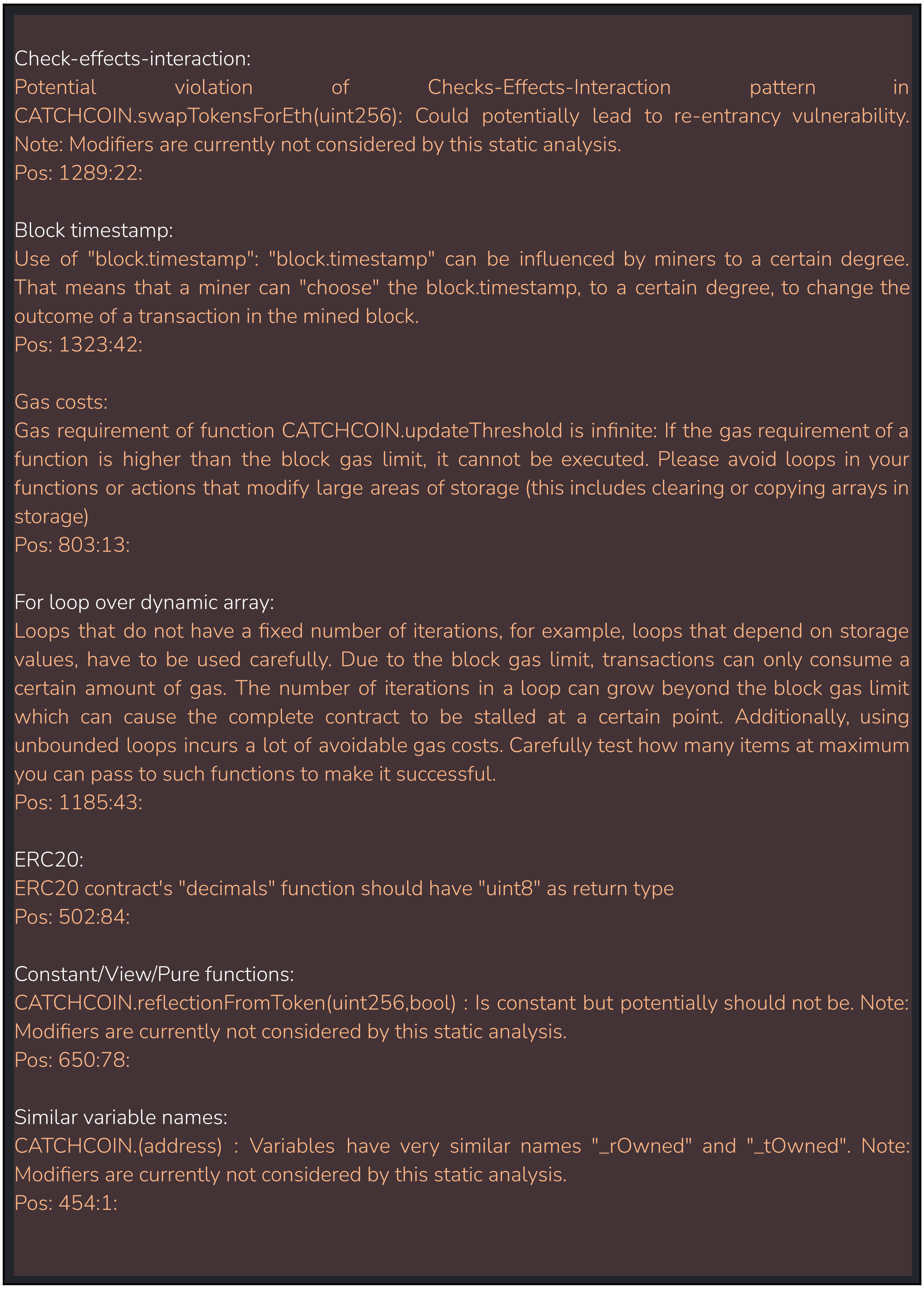
```
INFO:Detectors:
CATCHCOIN.allowance(address,address).owner (CatchCoin.sol#532) shadows:
	- Ownable.owner() (CatchCoin.sol#174-176) (function)
CATCHCOIN._approve(address,address,uint256).owner (CatchCoin.sol#1060) shadows:
	- Ownable.owner() (CatchCoin.sol#174-176) (function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in CATCHCOIN.swapAndLiquify(uint256) (CatchCoin.sol#1237-1258):
	External calls:
	- swapTokensForEth(half) (CatchCoin.sol#1249)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pat
h,address(this),block.timestamp) (CatchCoin.sol#1278-1284)
	- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
		- uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CatchCoin.sol#1303-1310)
	External calls sending eth:
	- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
		- uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CatchCoin.sol#1303-1310)
	State variables written after the call(s):
	- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
		- _allowances[owner][spender] = amount (CatchCoin.sol#1064)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CATCHCOIN.swapAndLiquify(uint256) (CatchCoin.sol#1237-1258):
	External calls:
	- swapTokensForEth(half) (CatchCoin.sol#1249)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,pat
h,address(this),block.timestamp) (CatchCoin.sol#1278-1284)
	- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
		- uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CatchCoin.sol#1303-1310)
	External calls sending eth:
	- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
		- uniswapV2Router.addLiquidityETH{value:
```

```
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(CatchCoin.sol#1303-1310)
	Event emitted after the call(s):
	- Approval(owner,spender,amount) (CatchCoin.sol#1065)
		- addLiquidity(otherHalf,newBalance) (CatchCoin.sol#1255)
	- SwapAndLiquify(half,newBalance,otherHalf) (CatchCoin.sol#1257)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
CATCHCOIN._transfer(address,address,uint256) (CatchCoin.sol#1101-1157) uses timestamp for
comparisons
	Dangerous comparisons:
	- startingHr >= block.timestamp (CatchCoin.sol#1139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
CATCHCOIN.includeInReward(address) (CatchCoin.sol#688-699) has costly operations inside a
loop:
	- _excluded.pop() (CatchCoin.sol#695)
CATCHCOIN.removeAllFee() (CatchCoin.sol#1031-1036) has costly operations inside a loop:
	- refAmt = 0 (CatchCoin.sol#1032)
CATCHCOIN.removeAllFee() (CatchCoin.sol#1031-1036) has costly operations inside a loop:
	- coinOperation = 0 (CatchCoin.sol#1033)
CATCHCOIN.removeAllFee() (CatchCoin.sol#1031-1036) has costly operations inside a loop:
	- liquidty = 0 (CatchCoin.sol#1034)
CATCHCOIN.removeAllFee() (CatchCoin.sol#1031-1036) has costly operations inside a loop:
	- burn = 0 (CatchCoin.sol#1035)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CatchCoin.sol#811-821) has costly
operations inside a loop:
	- _rTotal = _rTotal - rFee - rBurn (CatchCoin.sol#815)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CatchCoin.sol#811-821) has costly
operations inside a loop:
	- _tFeeTotal = _tFeeTotal + tFee (CatchCoin.sol#816)
CATCHCOIN._reflectFee(uint256,uint256,uint256) (CatchCoin.sol#811-821) has costly
operations inside a loop:
	- _tTotal = _tTotal - tBurn (CatchCoin.sol#818)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
CATCHCOIN._rTotal (CatchCoin.sol#384) 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
INFO:Detectors:
Function IUniswapV2Router01.WETH() (CatchCoin.sol#238) is not in mixedCase
Parameter CATCHCOIN.setFundWallet1(address)._fundWallet (CatchCoin.sol#761) is not in
mixedCase
Parameter CATCHCOIN.setSwapAndLiquifyEnabled(bool)._enabled (CatchCoin.sol#773) is not in
mixedCase
```

```
Parameter CATCHCOIN.updateThreshold(uint256)._amount (CatchCoin.sol#788) is not in
mixedCase
Parameter CATCHCOIN.calculateTaxFee(uint256)._amount (CatchCoin.sol#984) is not in
mixedCase
Parameter CATCHCOIN.calculateLiquidityFee(uint256)._amount (CatchCoin.sol#995) is not in
mixedCase
Parameter CATCHCOIN.calculateCoinOperartionTax(uint256)._amount (CatchCoin.sol#1006) is
not in mixedCase
Parameter CATCHCOIN.calculateBurnTax(uint256)._amount (CatchCoin.sol#1019) is not in
mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-c
onventions
INFO:Detectors:
Redundant expression "this (CatchCoin.sol#138)" inContext (CatchCoin.sol#132-141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
CATCHCOIN.slitherConstructorVariables() (CatchCoin.sol#372-1412) uses literals with too many
digits:
	- _tTotal = 180000000 * 10 ** 18 (CatchCoin.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Loop condition i < _excluded.length (CatchCoin.sol#953) should use cached array length instead
of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
CATCHCOIN.startingHr (CatchCoin.sol#392) should be immutable
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-decl
ared-immutable
INFO:Slither:CatchCoin.sol analyzed (7 contracts with 93 detectors), 33 result(s) found
```

Solidity Static Analysis

CatchCoin.sol



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: 1289:22:

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: 1323:42:

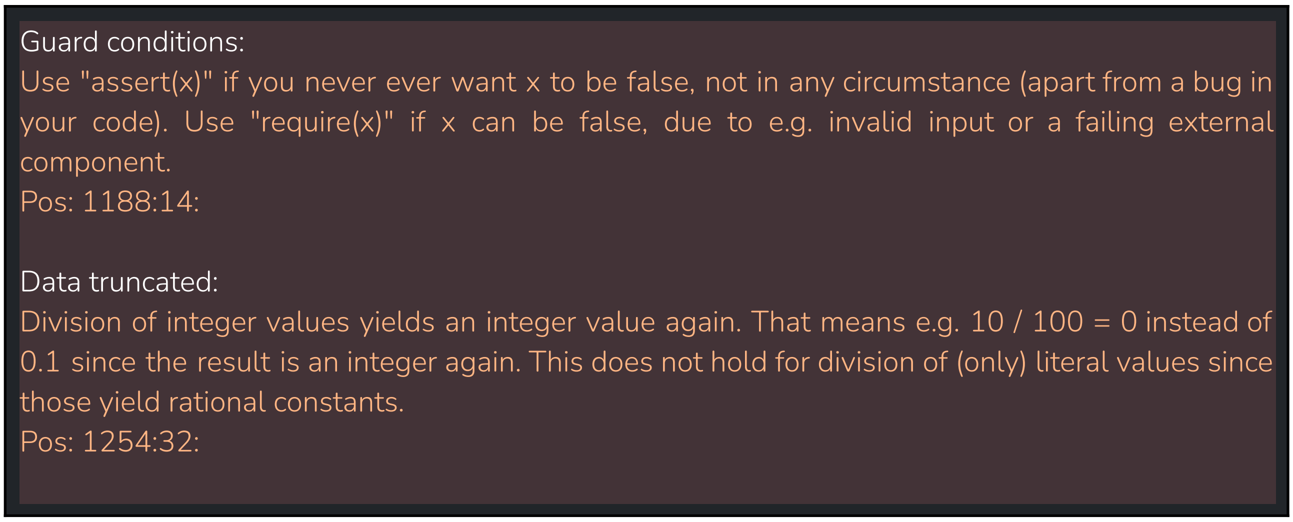
Gas costs:
Gas requirement of function CATCHCOIN.updateThreshold is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 803:13:

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: 1185:43:

ERC20:
ERC20 contract's "decimals" function should have "uint8" as return type
Pos: 502:84:

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: 650:78:

Similar variable names:
CATCHCOIN.(address) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.
Pos: 454:1:

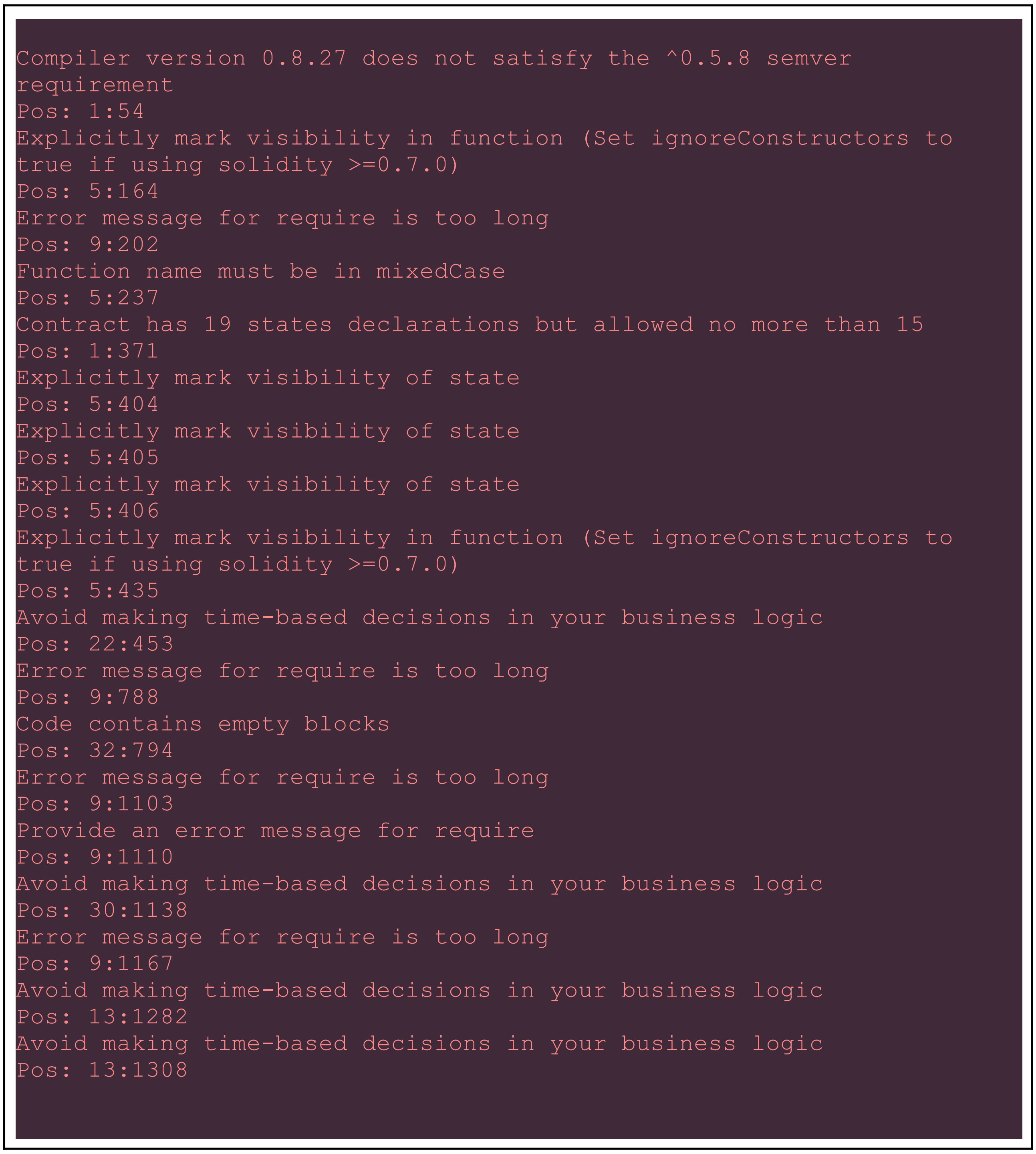


Guard conditions:
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
Pos: 1188:14:

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: 1254:32:

Solhint Linter

CatchCoin.sol



```
Compiler version 0.8.27 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:54
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:164
Error message for require is too long
Pos: 9:202
Function name must be in mixedCase
Pos: 5:237
Contract has 19 states declarations but allowed no more than 15
Pos: 1:371
Explicitly mark visibility of state
Pos: 5:404
Explicitly mark visibility of state
Pos: 5:405
Explicitly mark visibility of state
Pos: 5:406
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:435
Avoid making time-based decisions in your business logic
Pos: 22:453
Error message for require is too long
Pos: 9:788
Code contains empty blocks
Pos: 32:794
Error message for require is too long
Pos: 9:1103
Provide an error message for require
Pos: 9:1110
Avoid making time-based decisions in your business logic
Pos: 30:1138
Error message for require is too long
Pos: 9:1167
Avoid making time-based decisions in your business logic
Pos: 13:1282
Avoid making time-based decisions in your business logic
Pos: 13:1308
```

Software analysis result:

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

Ether Authority