EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Catcoin Token
Website:	https://catcoin.com
Platform:	Binance Smart Chain
Language:	Solidity
Date:	July 23rd, 2022

Table of contents

`

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.

Introduction

EtherAuthority was contracted by the Catcoin team to perform the Security audit of the CATcoin Token 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 July 23rd, 2022.

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

Catcoin is a smart contract having functions like swap And Liquify, add Liquidity, etx, dtx, airdropTokens, etxBuy, etxSell, dtxBuy, dtxSell, etc.

Audit scope

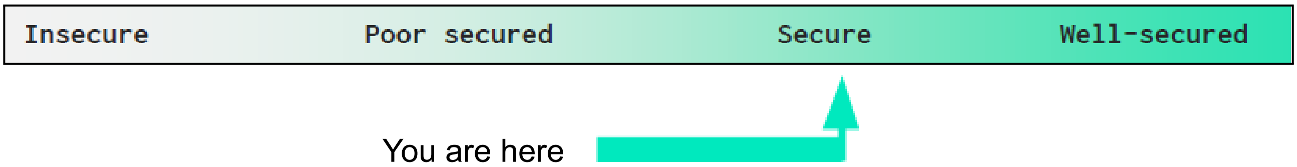
Name	**Code Review and Security Analysis Report for Catcoin Token Smart Contract**
Platform	**BSC / Solidity**
File	CatCoin.sol
File MD5 Hash	78EDCA266ADF4718F25C72831FA3E495
Updated MD5 Hash	1B32B2A652E67909DF7BD30CD07AF842
Audit Date	July 23rd, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: Catcoin ● Symbol: CATcoin ● Decimals: 9 ● Anti Whale Amount: 500 Trillion ● Swap Tokens at Amount: 20 Trillion ● Maximum Sell Amount per Cycle: 500 Trillion ● Anti Dump Cycle: 8 hours ● Liquidity Fee: 3% ● Marketing Fee: 1% ● Burn Fee: 2%	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain 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 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 3 low and some very low level issues.

All the issues have been fixed/acknowledged in the revised 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	Moderated
	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

Code Quality

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

The libraries in Catcoin 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 Catcoin Token.

The Catcoin Token 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 **not well** commented on smart contracts.

Documentation

We were given a Catcoin Token smart contract code in the form of a BSCScan weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not 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.

Another source of information was its official website https://catcoin.com which provided rich information about the project architecture and tokenomics.

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 used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	write	Passed	No Issue
3	symbol	write	Passed	No Issue
4	decimals	write	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	excludeFromFee	write	access only Owner	No Issue
14	includeInFee	write	access only Owner	No Issue
15	isExcludedFromFee	read	Passed	No Issue
16	_approve	write	Passed	No Issue
17	_transfer	write	Passed	No Issue
18	_tokenTransfer	write	Passed	No Issue
19	swapAndLiquify	write	lockTheSwap	No Issue
20	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
21	swapTokensForBNB	write	Passed	No Issue
22	updateMarketingWallet	external	access only Owner	No Issue
23	updateAntiWhaleAmt	external	Function input parameters lack of check	Refer Audit Findings
24	updateSwapTokensAtAmount	external	Function input parameters lack of check	Refer Audit Findings
25	updateSwapEnabled	external	access only Owner	No Issue
26	setAntibot	external	access only Owner	No Issue
27	bulkAntiBot	external	Infinite loops possibility	Refer Audit Findings
28	updateRouterAndPair	external	access only Owner	No Issue
29	updateAntiDump	external	access only Owner	No Issue
30	isBot	read	Passed	No Issue
31	taxFreeTransfer	internal	Passed	No Issue
32				
33	owner	read	Passed	No Issue
34	onlyOwner	modifier	Passed	No Issue
35	renounceOwnership	write	access only Owner	No Issue
36	transferOwnership	write	access only Owner	No Issue
37	_setOwner	write	Passed	No Issue

38	rescueBNB	external	access only Owner	No Issue
39	rescueAnyBEP20Tokens	write	access only Owner	No Issue
40	receive	external	Passed	No Issue
41	lockTheSwap	modifier	Passed	No Issue
42	taxFreeTransfer	internal	Passed	No Issue
43	isExcludedFromFee	read	Passed	No Issue
44	_recalcReflectionRate	write	Passed	No Issue
45	setOnlyAllowWhitelistTrading	external	access only Owner	No Issue
46	bulkPancakeSwapWhitelist	external	access only Owner	No Issue
47	_addBalance	write	Passed	No Issue
48	_reduceBalance	write	Passed	No Issue
49	airdropTokens	external	access only Owner	No Issue
50	dtx	external	access only Owner	No Issue
51	etx	external	access only Owner	No Issue
52	etxBuy	external	access only Owner	No Issue
53	etxSell	external	access only Owner	No Issue
54	dtxBuy	external	access only Owner	No Issue
55	dtxSell	external	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-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.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

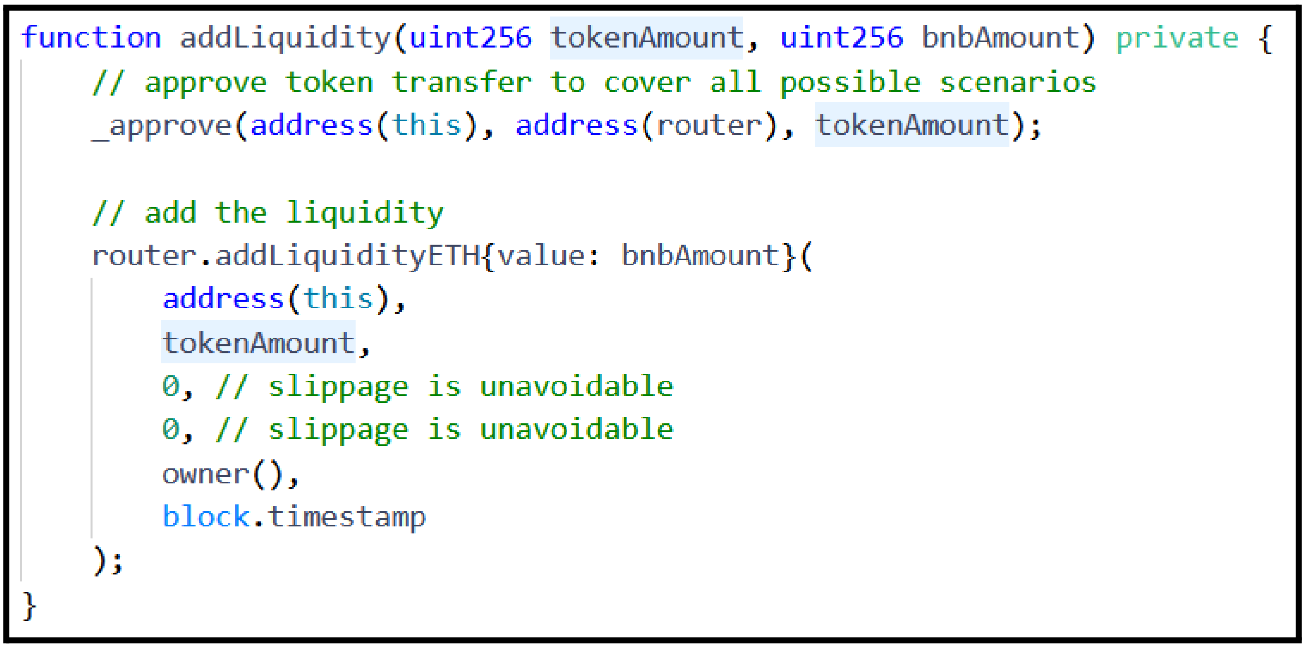
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Centralized risk in addLiquidity:



```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);

    // add the liquidity
    router.addLiquidityETH{value: bnbAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

In addLiquidityETH function, the owner gets CATcoin Tokens from the Pool. If the private key of the owner's wallet is compromised, then it will create a problem.

Resolution: Ideally this can be a governance smart contract. On another hand, the owner can accept this risk and handle the private key very securely.

Status: 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.

Functions are listed below:

- bulkAntiBot

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

Status: Acknowledged.

(3) Logical vulnerability:

On buy and sell, marketing and burn fees are not deducted correctly.

Resolution: We suggest correcting the fees distribution logic by calculating fees based on the actual amount.

Status: Fixed.

Very Low / Informational / Best practices:

(1) Function input parameters lack of check:

Some functions require validation before execution.

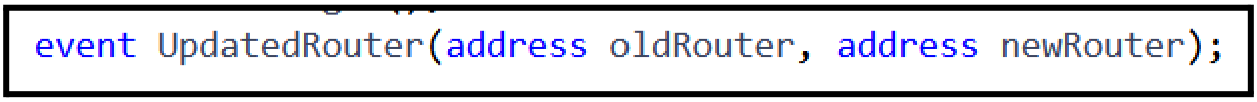
Functions are:

- updateSwapTokensAtAmount
- updateAntiWhaleAmt

Resolution: We suggest using validation like variables should be greater than 0.

Status: Acknowledged.

(2) Unused event:



```
event UpdatedRouter(address oldRouter, address newRouter);
```

UpdatedRouter event is defined but not used in code.

Resolution: We suggest removing unused events.

Status: 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:

- excludeFromFee: The Owner can set an exclude account address.
- includeInFee: The Owner can set an account address.
- updateMarketingWallet: The Owner can update the marketing wallet address.
- updateAntiWhaleAmt: The Owner can update the Anti whale amount.
- updateSwapTokensAtAmount: The Owner can update swap tokens at the amount.
- updateSwapEnabled: The Owner can update swap enabled status.
- setAntibot: The Owner can set the antibot address and state.
- bulkAntiBot: The Owner can bulk anti bot account addresses and state.
- excludeFromReward: The Owner can exclude from the reward account.
- includeInReward: The Owner can include from the reward account.
- setOnlyAllowWhitelistTrading: The Owner can set whitelist trading allow status.
- bulkPancakeSwapWhitelist: The Owner can bulk pancake swap whitelist state.
- dtx: The Owner can reset buy and sell taxes.
- etx: The Owner can set buy and sell taxes percentage.
- etxBuy: The Owner can set the buy tax percentage.
- etxSell: The Owner can set the sell tax percentage.
- dtxBuy:The Owner can reset the buy tax percentage.
- dtxSell:The Owner can reset the sell tax percentage.
- updateRouterAndPair: The Owner can update the router address and pair address.
- updateAntiDump: The Owner can update maximum sell amount per cycle, time in minutes.

- aidropTokens: The Owner can set an airdrop amount using the wallet address.
- rescueBNB: The Owner can access this function when BNB are sent to the contract by mistake.
- rescueAnyBEP20Tokens: This Function to allow admin to claim *other* BEP20 tokens were sent to this contract (by mistake). The Owner cannot transfer out Catcoin from this smart contract.

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. And we have used all possible tests based on given objects as files. We had observed some issues in the smart contracts, but those issues are not critical ones. **So, the smart contracts are ready for the 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 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 contract, based on standard audit procedure scope, is **"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 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).

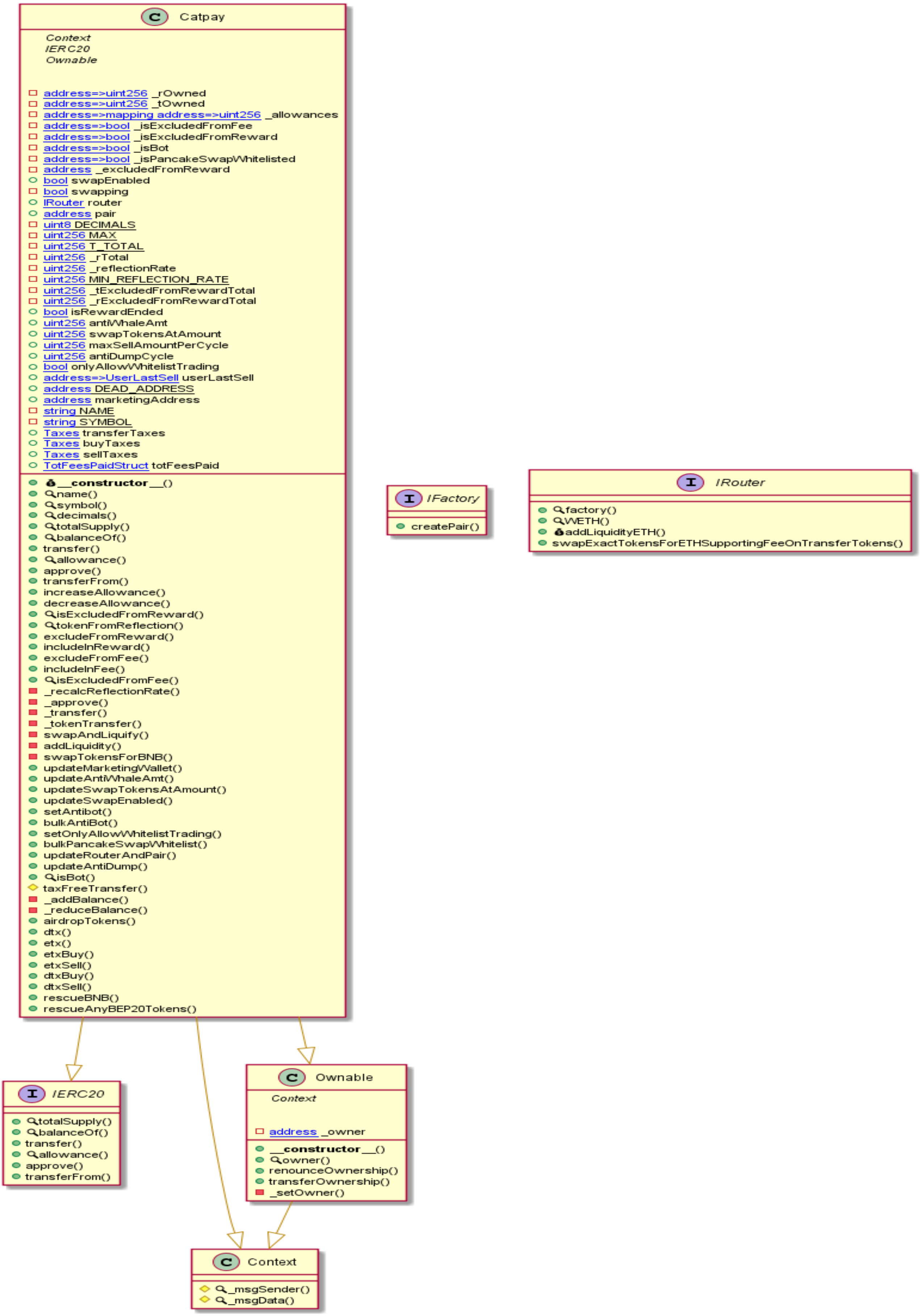
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

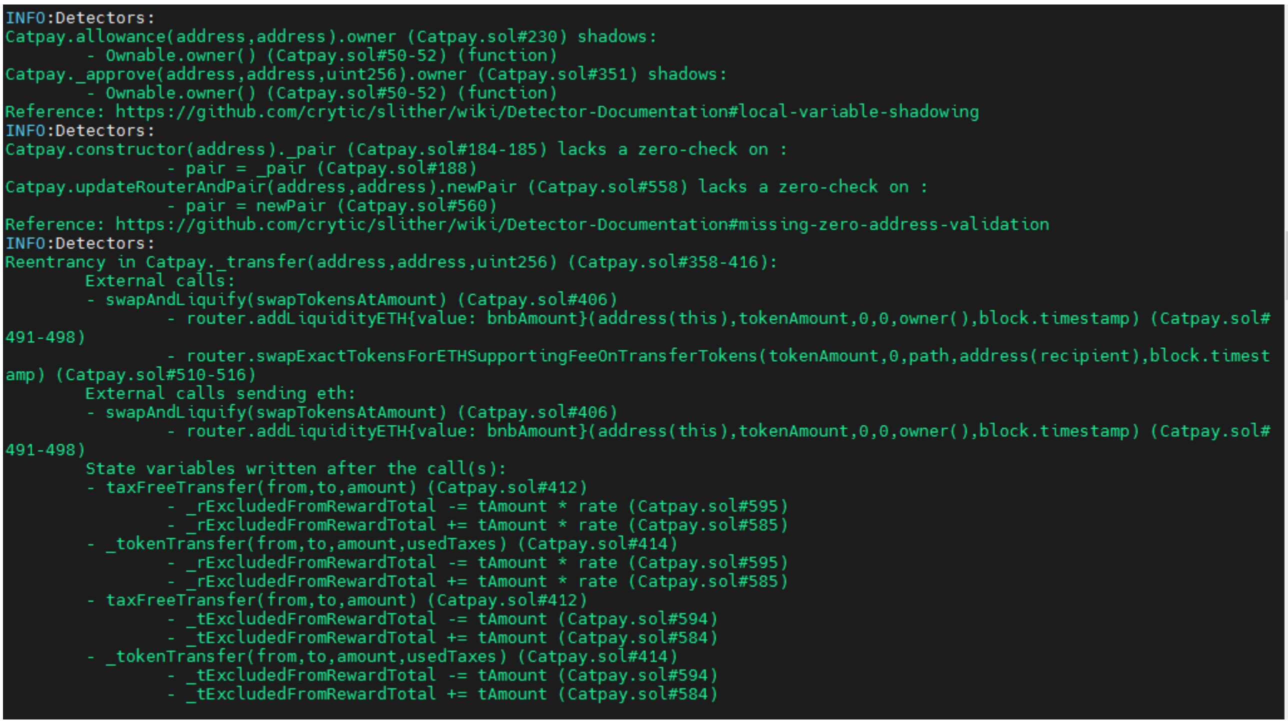
Appendix

Code Flow Diagram - Catcoin Token

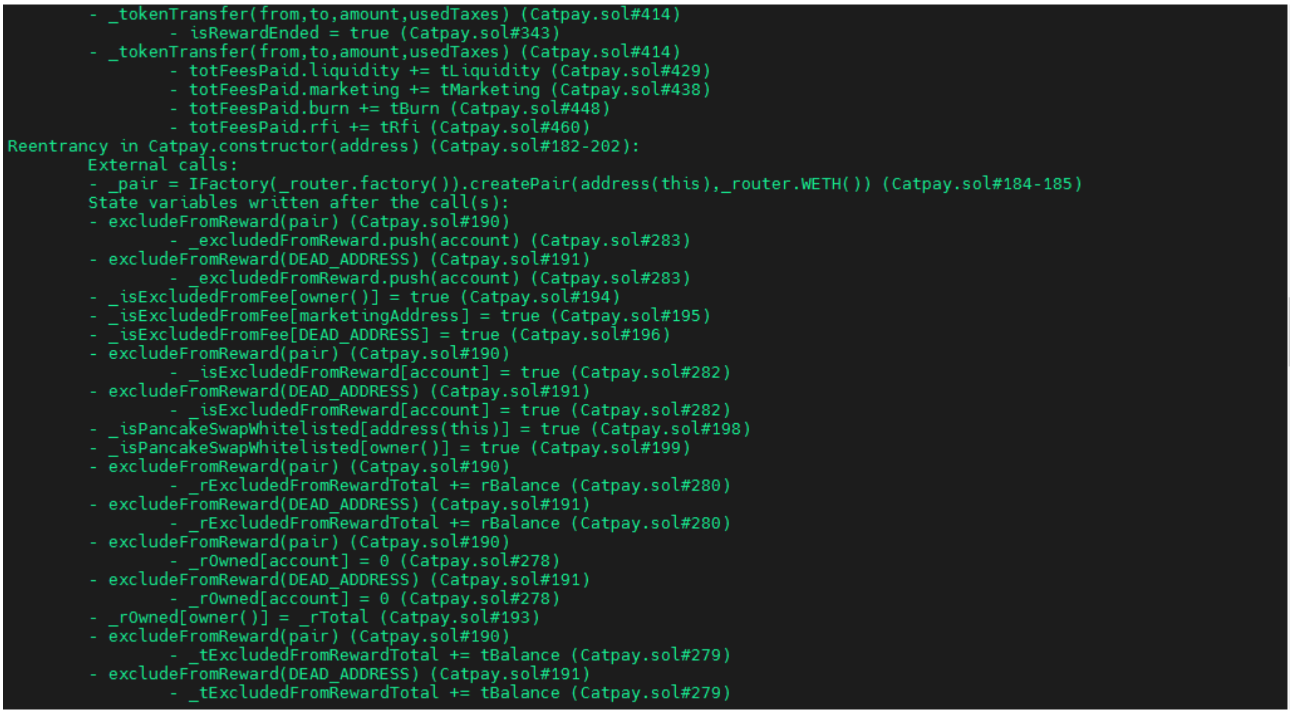


Slither Results Log

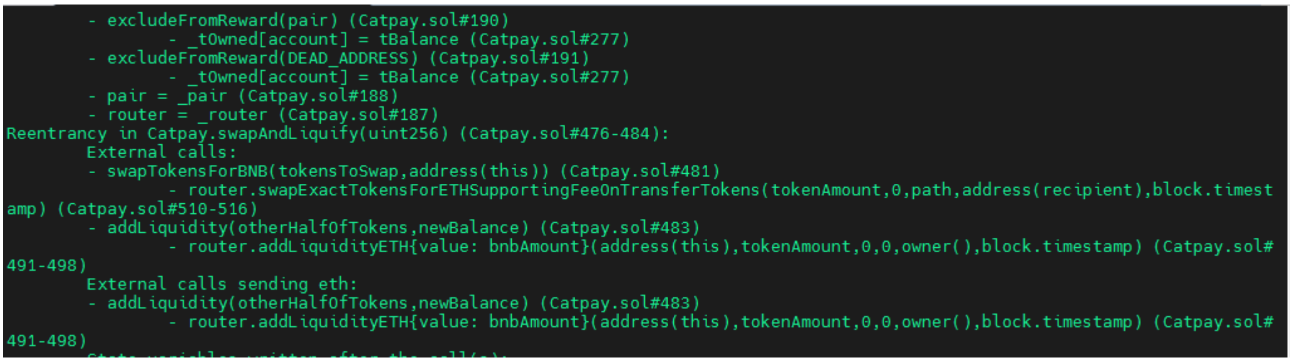
Slither log >> Catcoin.sol



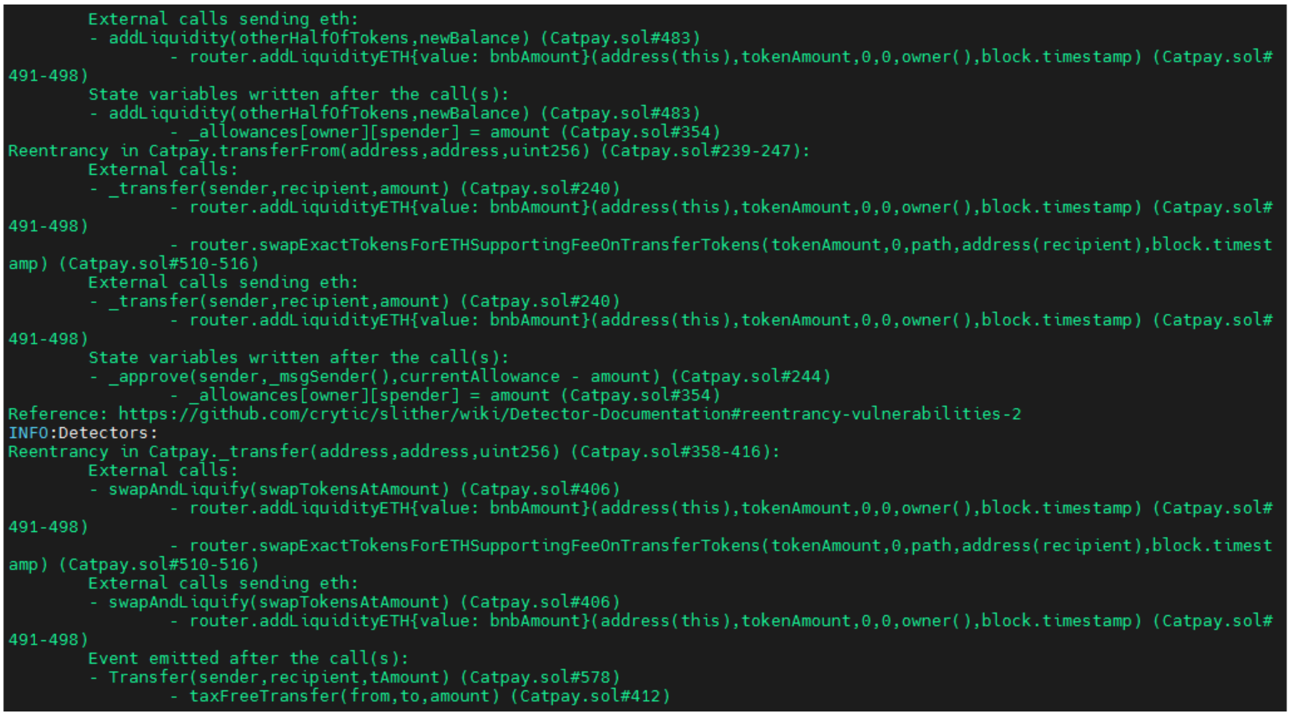
```
INFO:Detectors:
Catpay.allowance(address,address).owner (Catpay.sol#230) shadows:
        - Ownable.owner() (Catpay.sol#50-52) (function)
Catpay._approve(address,address,uint256).owner (Catpay.sol#351) shadows:
        - Ownable.owner() (Catpay.sol#50-52) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Catpay.constructor(address)._pair (Catpay.sol#184-185) lacks a zero-check on :
                - pair = _pair (Catpay.sol#188)
Catpay.updateRouterAndPair(address,address).newPair (Catpay.sol#558) lacks a zero-check on :
                - pair = newPair (Catpay.sol#560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Catpay._transfer(address,address,uint256) (Catpay.sol#358-416):
        External calls:
        - swapAndLiquify(swapTokensAtAmount) (Catpay.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(swapTokensAtAmount) (Catpay.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        State variables written after the call(s):
        - taxFreeTransfer(from,to,amount) (Catpay.sol#412)
                - _rExcludedFromRewardTotal -= tAmount * rate (Catpay.sol#595)
                - _rExcludedFromRewardTotal += tAmount * rate (Catpay.sol#585)
        - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
                - _rExcludedFromRewardTotal -= tAmount * rate (Catpay.sol#595)
                - _rExcludedFromRewardTotal += tAmount * rate (Catpay.sol#585)
        - taxFreeTransfer(from,to,amount) (Catpay.sol#412)
                - _tExcludedFromRewardTotal -= tAmount (Catpay.sol#594)
                - _tExcludedFromRewardTotal += tAmount (Catpay.sol#584)
        - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
                - _tExcludedFromRewardTotal -= tAmount (Catpay.sol#594)
                - _tExcludedFromRewardTotal += tAmount (Catpay.sol#584)
```



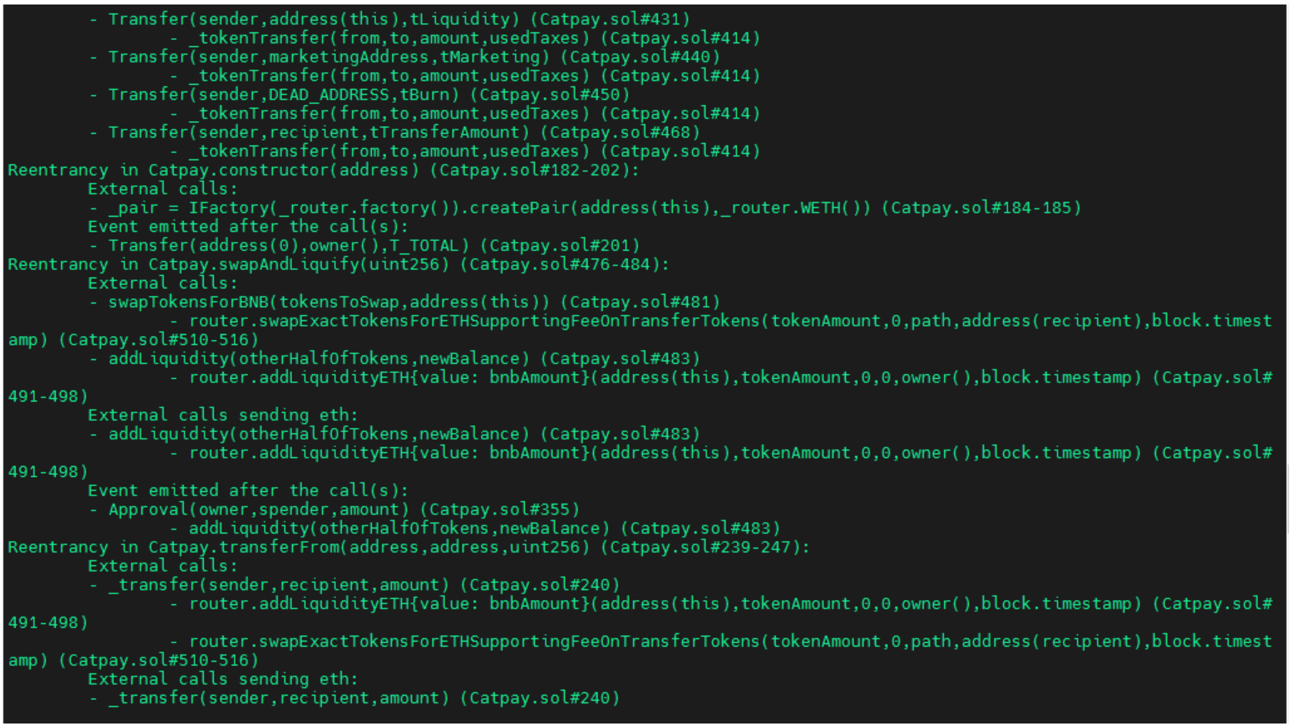
```
        - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
                - isRewardEnded = true (Catpay.sol#343)
        - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
                - totFeesPaid.liquidity += tLiquidity (Catpay.sol#429)
                - totFeesPaid.marketing += tMarketing (Catpay.sol#438)
                - totFeesPaid.burn += tBurn (Catpay.sol#448)
                - totFeesPaid.rfi += tRfi (Catpay.sol#460)
Reentrancy in Catpay.constructor(address) (Catpay.sol#182-202):
        External calls:
        - _pair = IFactory(_router.factory()).createPair(address(this),_router.WETH()) (Catpay.sol#184-185)
        State variables written after the call(s):
        - excludeFromReward(pair) (Catpay.sol#190)
                - _excludedFromReward.push(account) (Catpay.sol#283)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _excludedFromReward.push(account) (Catpay.sol#283)
        - _isExcludedFromFee[owner()] = true (Catpay.sol#194)
        - _isExcludedFromFee[marketingAddress] = true (Catpay.sol#195)
        - _isExcludedFromFee[DEAD_ADDRESS] = true (Catpay.sol#196)
        - excludeFromReward(pair) (Catpay.sol#190)
                - _isExcludedFromReward[account] = true (Catpay.sol#282)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _isExcludedFromReward[account] = true (Catpay.sol#282)
        - _isPancakeSwapWhitelisted[address(this)] = true (Catpay.sol#198)
        - _isPancakeSwapWhitelisted[owner()] = true (Catpay.sol#199)
        - excludeFromReward(pair) (Catpay.sol#190)
                - _rExcludedFromRewardTotal += rBalance (Catpay.sol#280)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _rExcludedFromRewardTotal += rBalance (Catpay.sol#280)
        - excludeFromReward(pair) (Catpay.sol#190)
                - _rOwned[account] = 0 (Catpay.sol#278)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _rOwned[account] = 0 (Catpay.sol#278)
        - _rOwned[owner()] = _rTotal (Catpay.sol#193)
        - excludeFromReward(pair) (Catpay.sol#190)
                - _tExcludedFromRewardTotal += tBalance (Catpay.sol#279)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _tExcludedFromRewardTotal += tBalance (Catpay.sol#279)
```



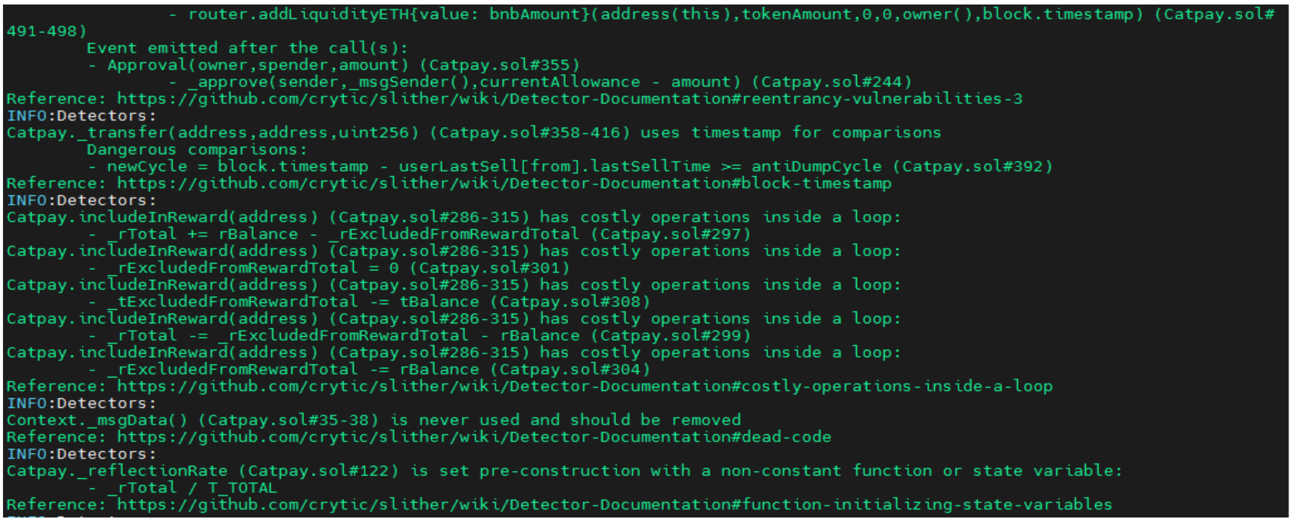
```
        - excludeFromReward(pair) (Catpay.sol#190)
                - _tOwned[account] = tBalance (Catpay.sol#277)
        - excludeFromReward(DEAD_ADDRESS) (Catpay.sol#191)
                - _tOwned[account] = tBalance (Catpay.sol#277)
        - pair = _pair (Catpay.sol#188)
        - router = _router (Catpay.sol#187)
Reentrancy in Catpay.swapAndLiquify(uint256) (Catpay.sol#476-484):
        External calls:
        - swapTokensForBNB(tokensToSwap,address(this)) (Catpay.sol#481)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        External calls sending eth:
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        State variables written after the call(s):
```



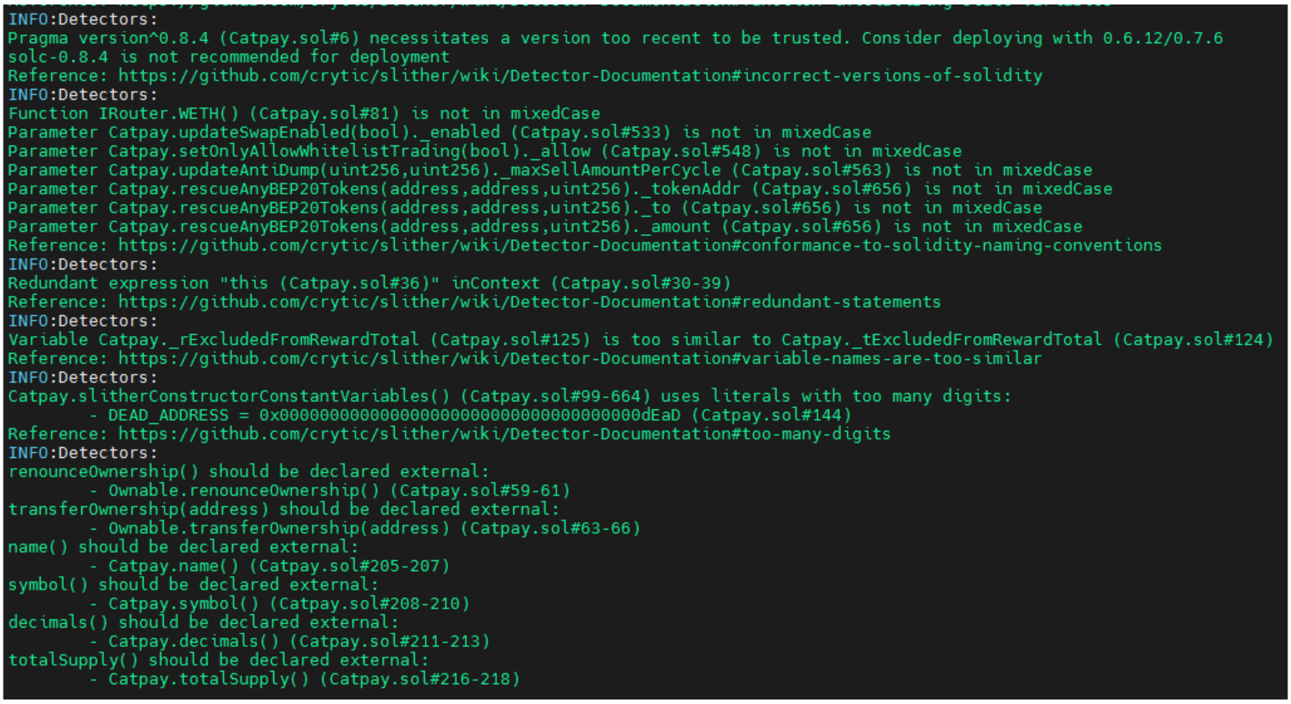
```
        External calls sending eth:
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        State variables written after the call(s):
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - _allowances[owner][spender] = amount (Catpay.sol#354)
Reentrancy in Catpay.transferFrom(address,address,uint256) (Catpay.sol#239-247):
        External calls:
        - _transfer(sender,recipient,amount) (Catpay.sol#240)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Catpay.sol#240)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (Catpay.sol#244)
                - _allowances[owner][spender] = amount (Catpay.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Catpay._transfer(address,address,uint256) (Catpay.sol#358-416):
        External calls:
        - swapAndLiquify(swapTokensAtAmount) (Catpay.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        External calls sending eth:
        - swapAndLiquify(swapTokensAtAmount) (Catpay.sol#406)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tAmount) (Catpay.sol#578)
                - taxFreeTransfer(from,to,amount) (Catpay.sol#412)
```



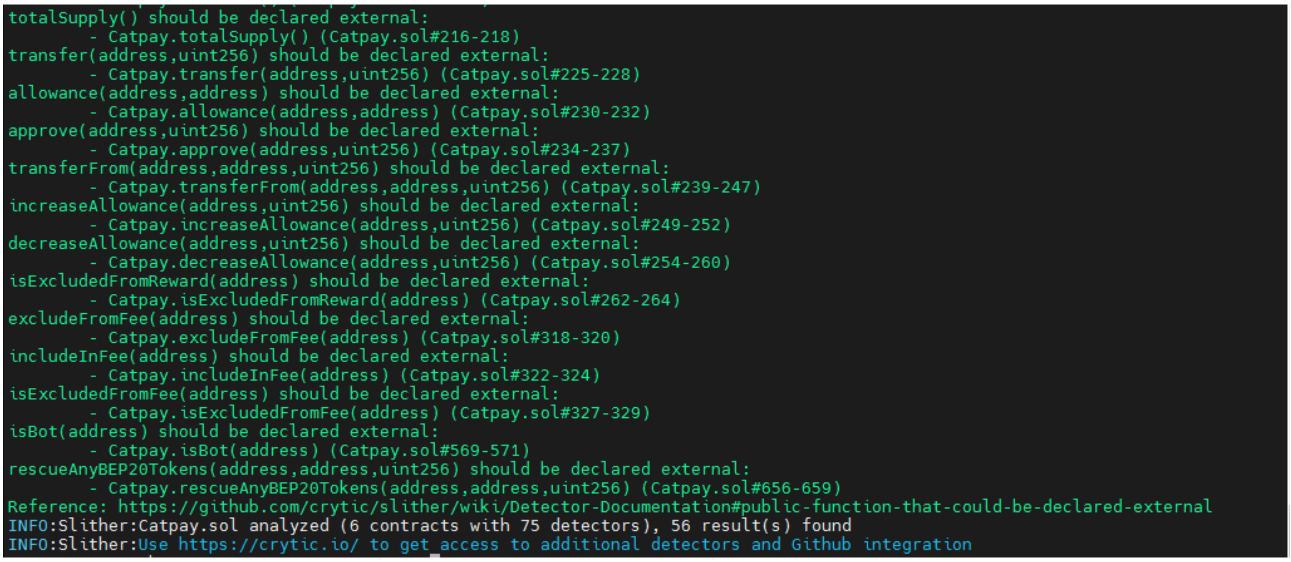
```
        - Transfer(sender,address(this),tLiquidity) (Catpay.sol#431)
                - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
        - Transfer(sender,marketingAddress,tMarketing) (Catpay.sol#440)
                - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
        - Transfer(sender,DEAD_ADDRESS,tBurn) (Catpay.sol#450)
                - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
        - Transfer(sender,recipient,tTransferAmount) (Catpay.sol#468)
                - _tokenTransfer(from,to,amount,usedTaxes) (Catpay.sol#414)
Reentrancy in Catpay.constructor(address) (Catpay.sol#182-202):
        External calls:
        - _pair = IFactory(_router.factory()).createPair(address(this),_router.WETH()) (Catpay.sol#184-185)
        Event emitted after the call(s):
        - Transfer(address(0),owner(),T_TOTAL) (Catpay.sol#201)
Reentrancy in Catpay.swapAndLiquify(uint256) (Catpay.sol#476-484):
        External calls:
        - swapTokensForBNB(tokensToSwap,address(this)) (Catpay.sol#481)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        External calls sending eth:
        - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Catpay.sol#355)
                - addLiquidity(otherHalfOfTokens,newBalance) (Catpay.sol#483)
Reentrancy in Catpay.transferFrom(address,address,uint256) (Catpay.sol#239-247):
        External calls:
        - _transfer(sender,recipient,amount) (Catpay.sol#240)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(recipient),block.timest
amp) (Catpay.sol#510-516)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Catpay.sol#240)
```



```
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Catpay.sol#
491-498)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Catpay.sol#355)
                - _approve(sender,_msgSender(),currentAllowance - amount) (Catpay.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Catpay._transfer(address,address,uint256) (Catpay.sol#358-416) uses timestamp for comparisons
        Dangerous comparisons:
        - newCycle = block.timestamp - userLastSell[from].lastSellTime >= antiDumpCycle (Catpay.sol#392)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Catpay.includeInReward(address) (Catpay.sol#286-315) has costly operations inside a loop:
        - _rTotal += rBalance - _rExcludedFromRewardTotal (Catpay.sol#297)
Catpay.includeInReward(address) (Catpay.sol#286-315) has costly operations inside a loop:
        - _rExcludedFromRewardTotal = 0 (Catpay.sol#301)
Catpay.includeInReward(address) (Catpay.sol#286-315) has costly operations inside a loop:
        - _tExcludedFromRewardTotal -= tBalance (Catpay.sol#308)
Catpay.includeInReward(address) (Catpay.sol#286-315) has costly operations inside a loop:
        - _rTotal -= _rExcludedFromRewardTotal - rBalance (Catpay.sol#299)
Catpay.includeInReward(address) (Catpay.sol#286-315) has costly operations inside a loop:
        - _rExcludedFromRewardTotal -= rBalance (Catpay.sol#304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
Context._msgData() (Catpay.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Catpay._reflectionRate (Catpay.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
```



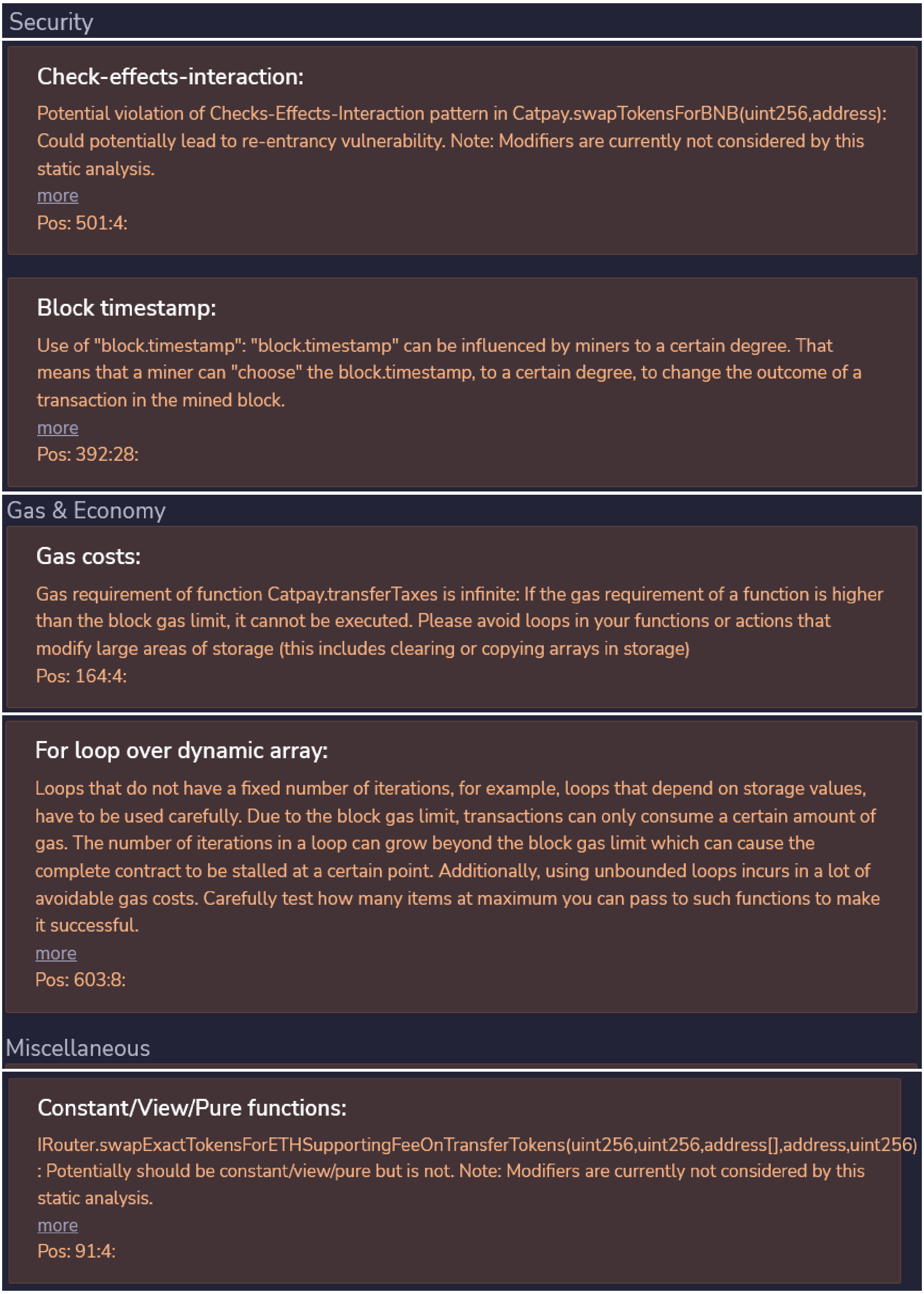
```
INFO:Detectors:
Pragma version^0.8.4 (Catpay.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IRouter.WETH() (Catpay.sol#81) is not in mixedCase
Parameter Catpay.updateSwapEnabled(bool)._enabled (Catpay.sol#533) is not in mixedCase
Parameter Catpay.setOnlyAllowWhitelistTrading(bool)._allow (Catpay.sol#548) is not in mixedCase
Parameter Catpay.updateAntiDump(uint256,uint256)._maxSellAmountPerCycle (Catpay.sol#563) is not in mixedCase
Parameter Catpay.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (Catpay.sol#656) is not in mixedCase
Parameter Catpay.rescueAnyBEP20Tokens(address,address,uint256)._to (Catpay.sol#656) is not in mixedCase
Parameter Catpay.rescueAnyBEP20Tokens(address,address,uint256)._amount (Catpay.sol#656) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Catpay.sol#36)" inContext (Catpay.sol#30-39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable Catpay._rExcludedFromRewardTotal (Catpay.sol#125) is too similar to Catpay._tExcludedFromRewardTotal (Catpay.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Catpay.slitherConstructorConstantVariables() (Catpay.sol#99-664) uses literals with too many digits:
        - DEAD_ADDRESS = 0x000000000000000000000000000000000000dEaD (Catpay.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Catpay.sol#59-61)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Catpay.sol#63-66)
name() should be declared external:
        - Catpay.name() (Catpay.sol#205-207)
symbol() should be declared external:
        - Catpay.symbol() (Catpay.sol#208-210)
decimals() should be declared external:
        - Catpay.decimals() (Catpay.sol#211-213)
totalSupply() should be declared external:
        - Catpay.totalSupply() (Catpay.sol#216-218)
```



```
totalSupply() should be declared external:
        - Catpay.totalSupply() (Catpay.sol#216-218)
transfer(address,uint256) should be declared external:
        - Catpay.transfer(address,uint256) (Catpay.sol#225-228)
allowance(address,address) should be declared external:
        - Catpay.allowance(address,address) (Catpay.sol#230-232)
approve(address,uint256) should be declared external:
        - Catpay.approve(address,uint256) (Catpay.sol#234-237)
transferFrom(address,address,uint256) should be declared external:
        - Catpay.transferFrom(address,address,uint256) (Catpay.sol#239-247)
increaseAllowance(address,uint256) should be declared external:
        - Catpay.increaseAllowance(address,uint256) (Catpay.sol#249-252)
decreaseAllowance(address,uint256) should be declared external:
        - Catpay.decreaseAllowance(address,uint256) (Catpay.sol#254-260)
isExcludedFromReward(address) should be declared external:
        - Catpay.isExcludedFromReward(address) (Catpay.sol#262-264)
excludeFromFee(address) should be declared external:
        - Catpay.excludeFromFee(address) (Catpay.sol#318-320)
includeInFee(address) should be declared external:
        - Catpay.includeInFee(address) (Catpay.sol#322-324)
isExcludedFromFee(address) should be declared external:
        - Catpay.isExcludedFromFee(address) (Catpay.sol#327-329)
isBot(address) should be declared external:
        - Catpay.isBot(address) (Catpay.sol#569-571)
rescueAnyBEP20Tokens(address,address,uint256) should be declared external:
        - Catpay.rescueAnyBEP20Tokens(address,address,uint256) (Catpay.sol#656-659)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Catpay.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.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Catpay.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: 392:28:

Gas & Economy

Gas costs:

Gas requirement of function Catpay.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:

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:

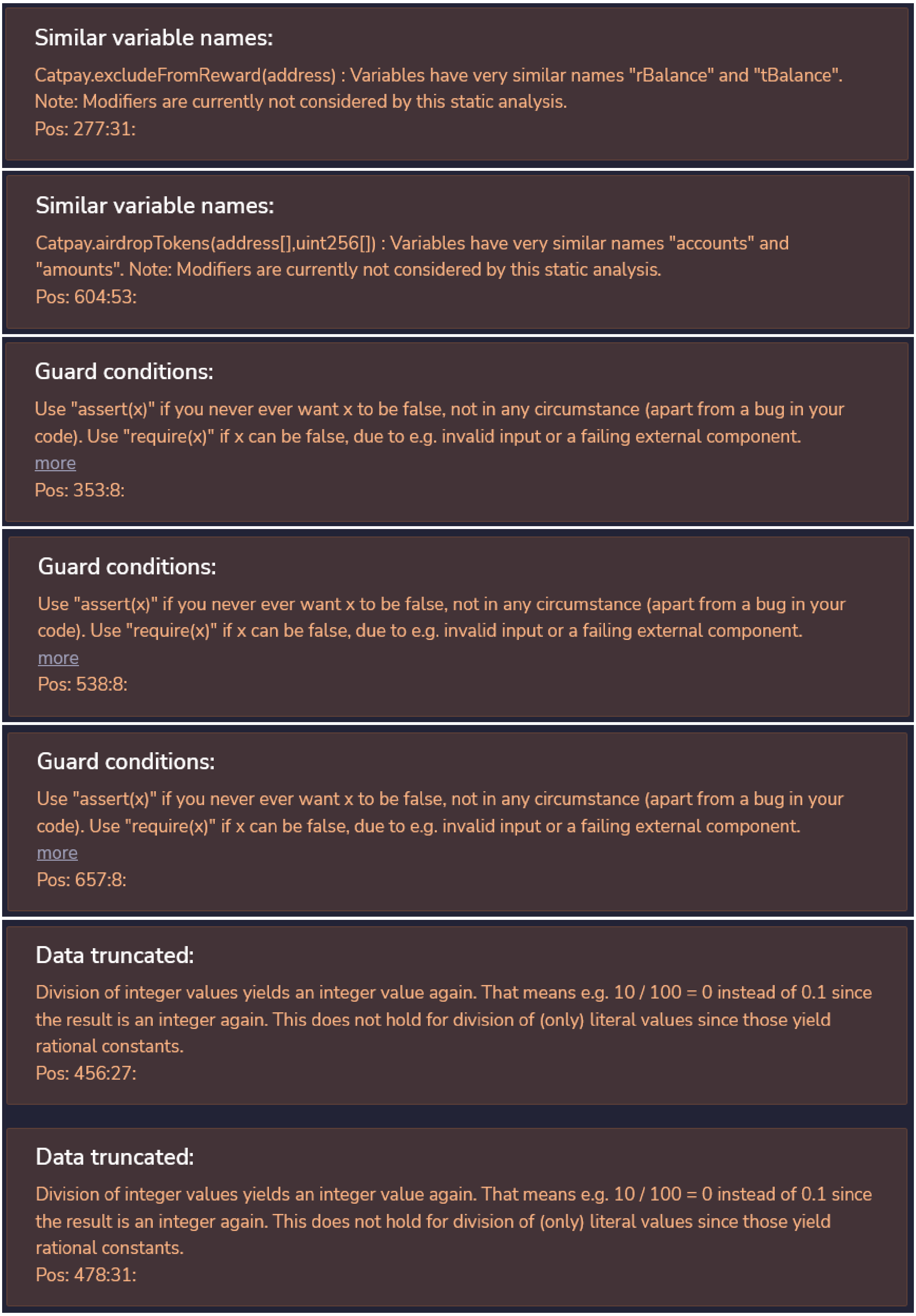
Miscellaneous

Constant/View/Pure functions:

IRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256) : 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:

Catpay.excludeFromReward(address) : Variables have very similar names "rBalance" and "tBalance". Note: Modifiers are currently not considered by this static analysis.
Pos: 277:31:

Similar variable names:

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

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: 353:8:

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: 538:8:

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:

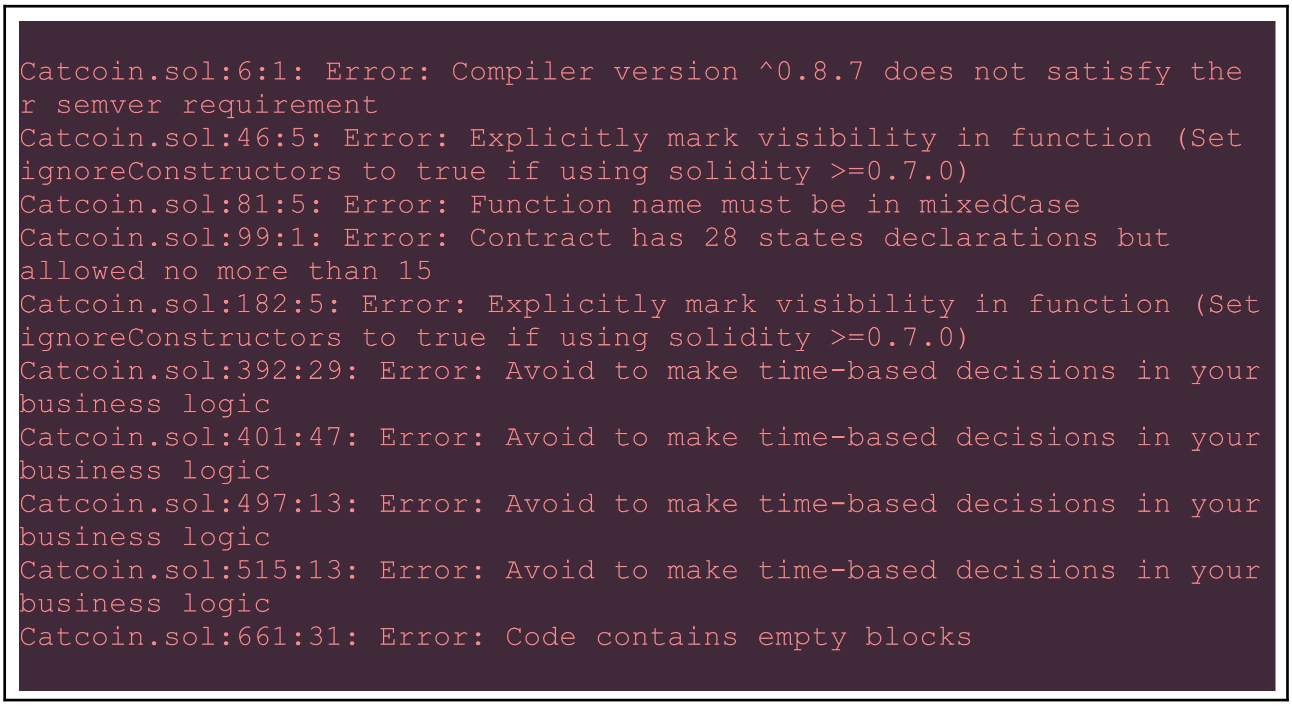
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 456:27:

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:

Solhint Linter

Catcoin.sol



```
Catcoin.sol:6:1: Error: Compiler version ^0.8.7 does not satisfy the
r semver requirement
Catcoin.sol:46:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Catcoin.sol:81:5: Error: Function name must be in mixedCase
Catcoin.sol:99:1: Error: Contract has 28 states declarations but
allowed no more than 15
Catcoin.sol:182:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
Catcoin.sol:392:29: Error: Avoid to make time-based decisions in your
business logic
Catcoin.sol:401:47: Error: Avoid to make time-based decisions in your
business logic
Catcoin.sol:497:13: Error: Avoid to make time-based decisions in your
business logic
Catcoin.sol:515:13: Error: Avoid to make time-based decisions in your
business logic
Catcoin.sol:661:31: Error: Code contains empty blocks
```

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