Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer: BrowniesSwap
Website: https://browniesswap.com
Platform: Binance Smart Chain
Language: Solidity
Date: October 22nd, 2021

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 BrowniesSwap team to perform the Security audit of the BrowniesSwap 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 October 17th, 2021.

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

BROWN Token is an BEP20 token smart contract, which acts as a backbone of the BrowniesSwap yields farm and AMM decentralized exchange. This audit only considers BROWN Token smart contract, and does not cover any other smart contracts in the platform.

Audit scope

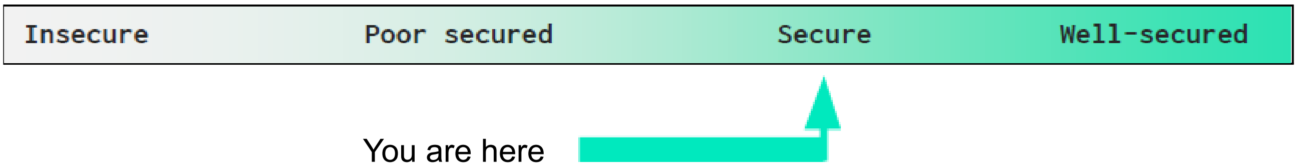
Name	**Code Review and Security Analysis Report for BrowniesSwap Token Smart Contract**
Platform	**BSC / Solidity**
File	CoinToken.sol
File MD5 Hash	1A93BAF4A2E55BAC1FB142355726A7F0
Online code	0x208fe37358d6aa767af66c4d87d5542ee2f35334
Audit Date	October 22nd, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: BrowniesSwap ● Symbol: BROWN ● Decimals: 9 ● Total Tokens: 1 Billion Tokens	**YES, This is valid.**
● Maximum Transaction Amount: 3 Million Tokens ● Minimum Tokens Before Swap: 0.2 Million Tokens ● BuyBack Upper Limit: 1 Billion Tokens	**YES, This is valid.**
● Tax Fee: 5% ● Marketing Fee: 4% ● BuyBack Fee: 3% ● Liquidity Fee: 7% (Marketing Fee + BuyBack Fee)	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

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 4 low and some very low level issues. These issues are not critical ones, so it's good to go for the production.

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	Moderated
	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	Moderated
	High consumption ‘for/while’ loop	Passed
	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 contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in BrowniesSwap 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 BrowniesSwap Token.

The BrowniesSwap 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 BrowniesSwap Token smart contracts code in the form of a BSCscan web link.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://browniesswap.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	lockTheSwap	modifier	Passed	No Issue
3	name	read	Passed	No Issue
4	symbol	read	Passed	No Issue
5	decimals	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	read	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	isExcludedFromReward	write	Passed	No Issue
15	totalFees	read	Passed	No Issue
16	minimumTokensBeforeSwapAmount	read	Passed	No Issue
17	buyBackUpperLimitAmount	read	Passed	No Issue
18	deliver	write	Critical operation lacks event log	Refer Audit Findings
19	reflectionFromToken	read	Passed	No Issue
20	tokenFromReflection	read	Passed	No Issue
21	excludeFromReward	write	Critical operation lacks event log	Refer Audit Findings
22	includeInReward	external	Infinite loops, Out of Gas issue	Refer Audit Findings
23	_approve	internal	Passed	No Issue
24	_transfer	internal	Passed	No Issue
25	swapTokens	internal	Passed	No Issue
26	buyBackTokens	internal	Passed	No Issue
27	swapTokensForEth	internal	Passed	No Issue
28	swapETHForTokens	internal	Passed	No Issue
29	addLiquidity	internal	Centralized risk in addLiquidity	Refer Audit Findings
30	_tokenTransfer	internal	Passed	No Issue
31	_transferStandard	internal	Passed	No Issue
32	_transferToExcluded	internal	Passed	No Issue
33	_transferFromExcluded	internal	Passed	No Issue
34	_transferBothExcluded	internal	Passed	No Issue
35	_reflectFee	internal	Passed	No Issue
36	_getValues	internal	Passed	No Issue
37	_getTValues	internal	Passed	No Issue
38	_getRValues	internal	Passed	No Issue

39	_getRate	internal	Passed	No Issue
40	_getCurrentSupply	internal	Infinite loops, Out of Gas issue	Refer Audit Findings
41	_takeLiquidity	internal	Passed	No Issue
42	calculateTaxFee	internal	Passed	No Issue
43	calculateLiquidityFee	internal	access only Architect	No Issue
44	removeAllFee	internal	Passed	No Issue
45	restoreAllFee	internal	Passed	No Issue
46	isExcludedFromFee	read	Passed	No Issue
47	excludeFromFee	write	Critical operation lacks event log	Refer Audit Findings
48	includeInFee	write	Critical operation lacks event log	Refer Audit Findings
49	setTaxFee	external	Critical operation lacks event log	Refer Audit Findings
50	setBuybackFee	external	Critical operation lacks event log	Refer Audit Findings
51	setMaxTxAmount	external	Critical operation lacks event log	Refer Audit Findings
52	setMarketingFee	external	Critical operation lacks event log	Refer Audit Findings
53	setNumTokensSellToAddToLiquidity	external	Critical operation lacks event log	Refer Audit Findings
54	setBuybackUpperLimit	external	Critical operation lacks event log	Refer Audit Findings
55	setMarketingAddress	external	Critical operation lacks event log	Refer Audit Findings
56	setSwapAndLiquifyEnabled	write	Critical operation lacks event log	Refer Audit Findings
57	setBuyBackEnabled	write	Critical operation lacks event log	Refer Audit Findings
58	presale	external	Critical operation lacks event log	Refer Audit Findings
59	transferToAddressETH	internal	Passed	No Issue
60	owner	read	Passed	No Issue
61	onlyOwner	modifier	Passed	No Issue
62	renounceOwnership	write	access only Owner	No Issue
63	transferOwnership	write	access only Owner	No Issue
64	geUnlockTime	read	Passed	No Issue
	getTime	read	Passed	No Issue
65	lock	write	access only Owner	No Issue
66	unlock	write	Ownership can be regained	Refer Audit Findings
67	_msgSender	read	Passed	No Issue
68	_msgData	read	Passed	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.

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) Critical operation lacks event log:

There are several places in the smart contracts, where a critical function call event log was not added. We suggest to add appropriate event log in the following functions:

- deliver()
- excludeFromReward()
- includeInReward()
- excludeFromFee()
- includeInFee()
- setTaxFee()
- setBuybackFee()
- setMaxTxAmount()
- setMarketingFee()
- setNumTokensSellToAddToLiquidity()
- setBuybackUpperLimit()
- setMarketingAddress()
- setSwapAndLiquifyEnabled()
- setBuyBackEnabled()
- presale()

(2) Possible to gain ownership:

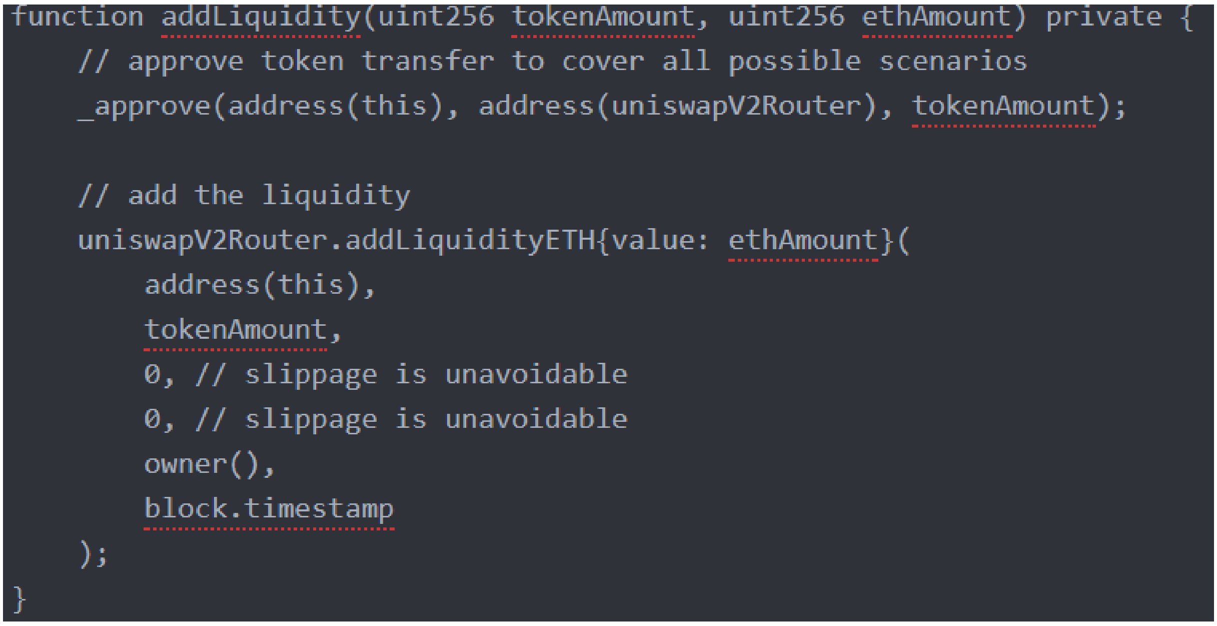
Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

1. Owner calls the lock function in contract to set the current owner as _previousOwner.
2. Owner calls unlock to unlock contract and set _owner = _previousOwner.
3. Owner called renounceOwnership to leave the contract without the owner.
4. Owner calls unlock to regain ownership.

Resolution: Remove these lock/unlock functions as this seems not serving a great purpose OR always renounce ownership first before calling the lock function

(3) Centralized risk in addLiquidity:

In addLiquidityETH function, owner gets BROWN Tokens from the Pool. If the private key of the owner wallet would be compromised, then it will create a problem.



```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

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.

(4) Infinite loops, Out of Gas issue:

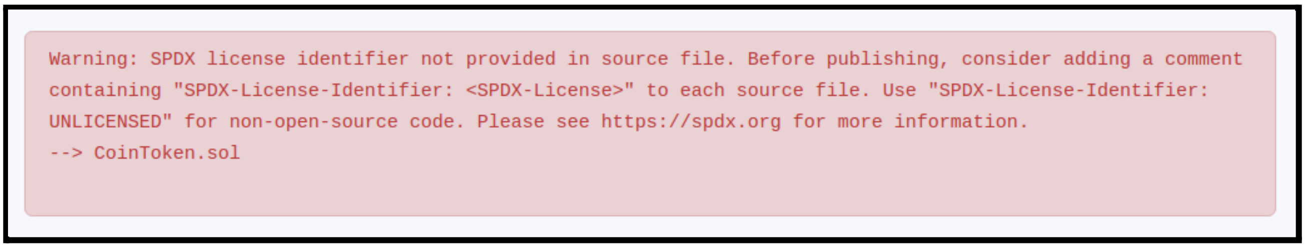
As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that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

Very Low / Informational / Best practices:

(1) Warning: SPDX license identifier:



```
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment
containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier:
UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> CoinToken.sol
```

Warning: SPDX license identifier not provided in source file.

Resolution: SPDX-License-Identifier.

(2) SafeMath Library:

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

(3) All functions which are not called internally, must be declared as external. It is more efficient as sometimes it saves some gas.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Centralization

These smart contracts have 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:

- excludeFromReward: Owner can check if the account is excluded or not.
- includeInReward: Owner can check includeInReward.
- excludeFromFee: Owner can check if the account is excludedFee or not.
- includeInFee: Owner can check includeInFee.
- setTaxFee: Owner can set Tax Fee Percent.
- setBuybackFee: Owner can set BuyBack Fee Percent.
- setMaxTxAmount: Owner can set Max Tx Amount.
- setSwapAndLiquifyEnabled: Owner can set Swap And Liquify Enabled.
- setMarketingFee: Owner can set Marketing Fee.
- setNumTokensSellToAddToLiquidity: Owner can set minimum tokens before swap to add liquidity.
- setBuybackUpperLimit: Owner can set BuyBack upper limit.
- setMarketingAddress: Owner can set marketing address.
- setBuyBackEnabled: Owner can set BuyBack Enabled.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones. So, **it's good to go to 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 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).

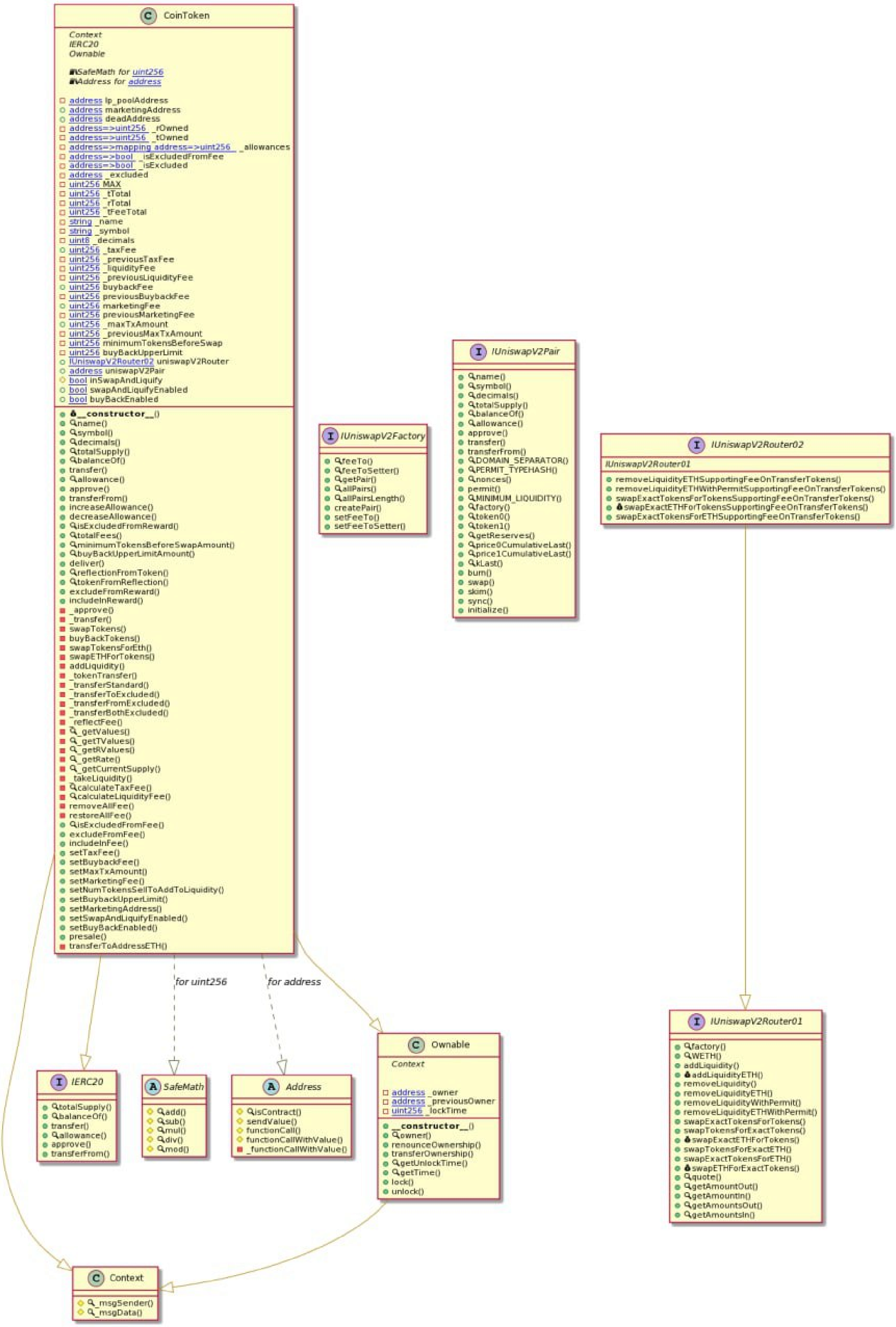
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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



Slither Results Log

Slither log >> CoinToken.sol

```
CoinToken.swapETHForTokens(uint256) (CoinToken.sol#737-752) sends eth to
arbitrary user
Dangerous calls:
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
CoinToken.addLiquidity(uint256,uint256) (CoinToken.sol#754-767) sends eth to
arbitrary user
Dangerous calls:
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp) (CoinToken.sol#759-766)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#functions-that-send-ether-to-arbitrary-destinations
Reentrancy in CoinToken._transfer(address,address,uint256) (CoinToken.sol#657-
695):
External calls:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
External calls sending eth:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- recipient.transfer(amount) (CoinToken.sol#978)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
State variables written after the call(s):
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _liquidityFee = _previousLiquidityFee (CoinToken.sol#906)
- _liquidityFee = 0 (CoinToken.sol#899)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _rOwned[address(this)] =_rOwned[address(this)].add(rLiquidity) (CoinToken.sol#873)
- _rOwned[sender] = _rOwned[sender].sub(rAmount) (CoinToken.sol#789)
- _rOwned[sender] = _rOwned[sender].sub(rAmount) (CoinToken.sol#798)
- _rOwned[sender] = _rOwned[sender].sub(rAmount) (CoinToken.sol#809)
- _rOwned[sender] = _rOwned[sender].sub(rAmount) (CoinToken.sol#819)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CoinToken.sol#790)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CoinToken.sol#800)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CoinToken.sol#810)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (CoinToken.sol#821)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _rTotal = _rTotal.sub(rFee) (CoinToken.sol#828)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _tOwned[address(this)] =_tOwned[address(this)].add(tLiquidity) (CoinToken.sol#875)
- _tOwned[sender] = _tOwned[sender].sub(tAmount) (CoinToken.sol#818)
- _tOwned[sender] = _tOwned[sender].sub(tAmount) (CoinToken.sol#808)
- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (CoinToken.sol#799)
- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (CoinToken.sol#820)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- inSwapAndLiquify = true (CoinToken.sol#485)
- inSwapAndLiquify = false (CoinToken.sol#487)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- marketingFee = previousMarketingFee (CoinToken.sol#908)
- marketingFee = 0 (CoinToken.sol#901)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address) (CoinToken.sol#490-530) performs a multiplication on the result of
a division:
-_maxTxAmount = _tTotal.div(1000).mul(3) (CoinToken.sol#510)
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address) (CoinToken.sol#490-530) performs a multiplication on the result of
a division:
-minimumTokensBeforeSwap = _tTotal.div(10000).mul(2) (CoinToken.sol#512)
CoinToken.swapTokens(uint256) (CoinToken.sol#697-708) performs a multiplication
on the result of a division:
- transferToAddressETH(lp_poolAddress,transferredBalance.div(_liquidityFee).mul(25
)) (CoinToken.sol#705)
CoinToken.swapTokens(uint256) (CoinToken.sol#697-708) performs a multiplication
on the result of a division:
- transferToAddressETH(marketingAddress,transferredBalance.div(_liquidityFee).mul(
marketingFee.sub(25))) (CoinToken.sol#706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dividebefore-
multiply
CoinToken.addLiquidity(uint256,uint256) (CoinToken.sol#754-767) ignores return
value by uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp) (CoinToken.sol#759-766)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unusedreturn
CoinToken.allowance(address,address).owner (CoinToken.sol#558) shadows:
- Ownable.owner() (CoinToken.sol#159-161) (function)
CoinToken._approve(address,address,uint256).owner (CoinToken.sol#649) shadows:
- Ownable.owner() (CoinToken.sol#159-161) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#localvariable-
shadowing
CoinToken.setTaxFee(uint256) (CoinToken.sol#923-925) should emit an event for:
- _taxFee = taxFee (CoinToken.sol#924)
CoinToken.setBuybackFee(uint256) (CoinToken.sol#927-930) should emit an event
for:
- buybackFee = _buybackFee (CoinToken.sol#928)
- _liquidityFee = buybackFee.add(marketingFee) (CoinToken.sol#929)
CoinToken.setMaxTxAmount(uint256) (CoinToken.sol#932-934) should emit an event
for:
- _maxTxAmount = maxTxAmount (CoinToken.sol#933)
CoinToken.setMarketingFee(uint256) (CoinToken.sol#936-939) should emit an event
for:
- marketingFee = _marketingFee (CoinToken.sol#937)
- _liquidityFee = buybackFee.add(marketingFee) (CoinToken.sol#938)
CoinToken.setNumTokensSellToAddToLiquidity(uint256) (CoinToken.sol#941-943)
should emit an event for:
- minimumTokensBeforeSwap = _minimumTokensBeforeSwap (CoinToken.sol#942)
```

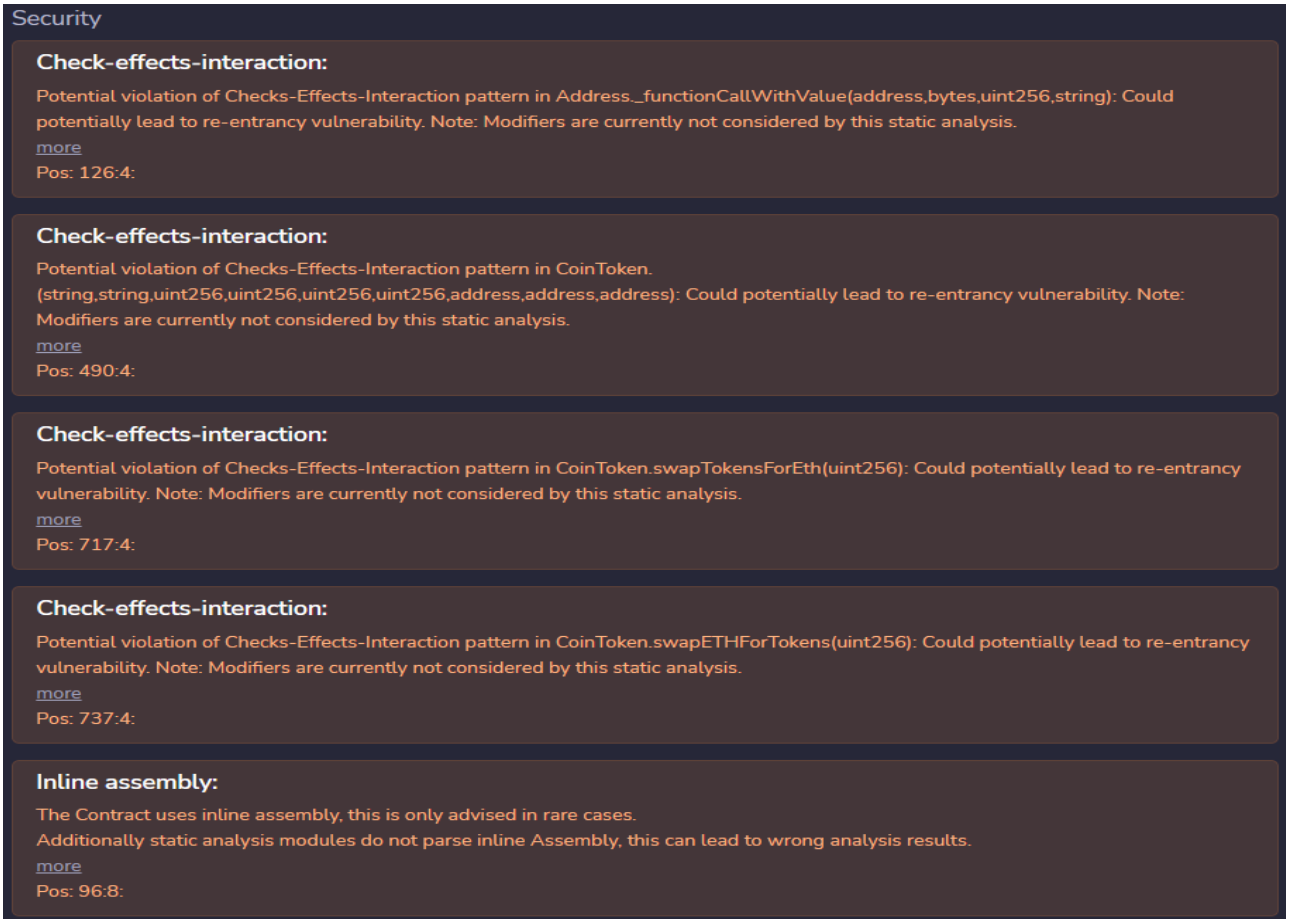
```
CoinToken.setBuybackUpperLimit(uint256) (CoinToken.sol#945-947) should emit an
event for:
- buyBackUpperLimit = buyBackLimit (CoinToken.sol#946)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#missing-events-arithmetic
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address)._ma (CoinToken.sol#490) lacks a zero-check on :
- marketingAddress = address(_ma) (CoinToken.sol#499)
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address)._lp (CoinToken.sol#490) lacks a zero-check on :
- lp_poolAddress = address(_lp) (CoinToken.sol#500)
- address(_lp).transfer(msg.value) (CoinToken.sol#527)
CoinToken.setMarketingAddress(address)._marketingAddress (CoinToken.sol#949)
lacks a zero-check on :
- marketingAddress = address(_marketingAddress)
(CoinToken.sol#950)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#missing-zero-address-validation
Reentrancy in CoinToken._transfer(address,address,uint256) (CoinToken.sol#657-
695):
External calls:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
External calls sending eth:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- recipient.transfer(amount) (CoinToken.sol#978)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
State variables written after the call(s):
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _previousLiquidityFee = _liquidityFee (CoinToken.sol#894)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _previousTaxFee = _taxFee (CoinToken.sol#893)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _tFeeTotal = _tFeeTotal.add(tFee) (CoinToken.sol#829)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- _taxFee = _previousTaxFee (CoinToken.sol#905)
- _taxFee = 0 (CoinToken.sol#898)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- buybackFee = previousBuybackFee (CoinToken.sol#907)
- buybackFee = 0 (CoinToken.sol#900)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- previousBuybackFee = buybackFee (CoinToken.sol#895)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- previousMarketingFee = marketingFee (CoinToken.sol#896)
Reentrancy in
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address) (CoinToken.sol#490-530):
External calls:
```

```
- uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV
2Router.WETH()) (CoinToken.sol#519-520)
State variables written after the call(s):
- _isExcludedFromFee[owner()] = true (CoinToken.sol#525)
- _isExcludedFromFee[address(this)] = true (CoinToken.sol#526)
- uniswapV2Router = _uniswapV2Router (CoinToken.sol#522)
Reentrancy in CoinToken.transferFrom(address,address,uint256)
(CoinToken.sol#567-571):
External calls:
- _transfer(sender,recipient,amount) (CoinToken.sol#568)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
-  uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
External calls sending eth:
- _transfer(sender,recipient,amount) (CoinToken.sol#568)
- recipient.transfer(amount) (CoinToken.sol#978)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
State variables written after the call(s):
- _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))
(CoinToken.sol#569)
- _allowances[owner][spender] = amount (CoinToken.sol#653)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities-2
Reentrancy in CoinToken._transfer(address,address,uint256) (CoinToken.sol#657-
695):
External calls:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
External calls sending eth:
- swapTokens(contractTokenBalance) (CoinToken.sol#675)
- recipient.transfer(amount) (CoinToken.sol#978)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
Event emitted after the call(s):
- SwapETHForTokens(amount,path) (CoinToken.sol#751)
- buyBackTokens(balance.div(100)) (CoinToken.sol#683)
- Transfer(sender,recipient,tTransferAmount) (CoinToken.sol#793)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- Transfer(sender,recipient,tTransferAmount) (CoinToken.sol#803)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- Transfer(sender,recipient,tTransferAmount) (CoinToken.sol#813)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
- Transfer(sender,recipient,tTransferAmount) (CoinToken.sol#824)
- _tokenTransfer(from,to,amount,takeFee) (CoinToken.sol#694)
Reentrancy in
```

```
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,address,addr
ess,address) (CoinToken.sol#490-530):
External calls:
- uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV
2Router.WETH()) (CoinToken.sol#519-520)
External calls sending eth:
- address(_lp).transfer(msg.value) (CoinToken.sol#527)
Event emitted after the call(s):
- Transfer(address(0),_msgSender(),_tTotal) (CoinToken.sol#529)
Reentrancy in CoinToken.swapETHForTokens(uint256) (CoinToken.sol#737-752):
External calls:
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
Event emitted after the call(s):
- SwapETHForTokens(amount,path) (CoinToken.sol#751)
Reentrancy in CoinToken.swapTokensForEth(uint256) (CoinToken.sol#717-735):
External calls:
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
Event emitted after the call(s):
- SwapTokensForETH(tokenAmount,path) (CoinToken.sol#734)
Reentrancy in CoinToken.transferFrom(address,address,uint256)
(CoinToken.sol#567-571):
External calls:
- _transfer(sender,recipient,amount) (CoinToken.sol#568)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(tokenAmount,0 ,path,address(this),block.timestamp) (CoinToken.sol#726-732)
External calls sending eth:
- _transfer(sender,recipient,amount) (CoinToken.sol#568)
- recipient.transfer(amount) (CoinToken.sol#978)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (CoinToken.sol#744-749)
Event emitted after the call(s):
- Approval(owner,spender,amount) (CoinToken.sol#654)
- _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))
(CoinToken.sol#569)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#reentrancy-vulnerabilities-3
Ownable.unlock() (CoinToken.sol#194-199) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp > _lockTime,Contract is locked
until 7 days) (CoinToken.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#blocktimestamp
Address.isContract(address) (CoinToken.sol#89-98) uses assembly
- INLINE ASM (CoinToken.sol#96)
Address._functionCallWithValue(address,bytes,uint256,string) (CoinToken.sol#126-
143) uses assembly
- INLINE ASM (CoinToken.sol#135-138)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#assembly-usage
```

Solidity Static Analysis

CoinToken.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 126:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CoinToken.(string,string,uint256,uint256,uint256,uint256,address,address,address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 490:4:

Check-effects-interaction:

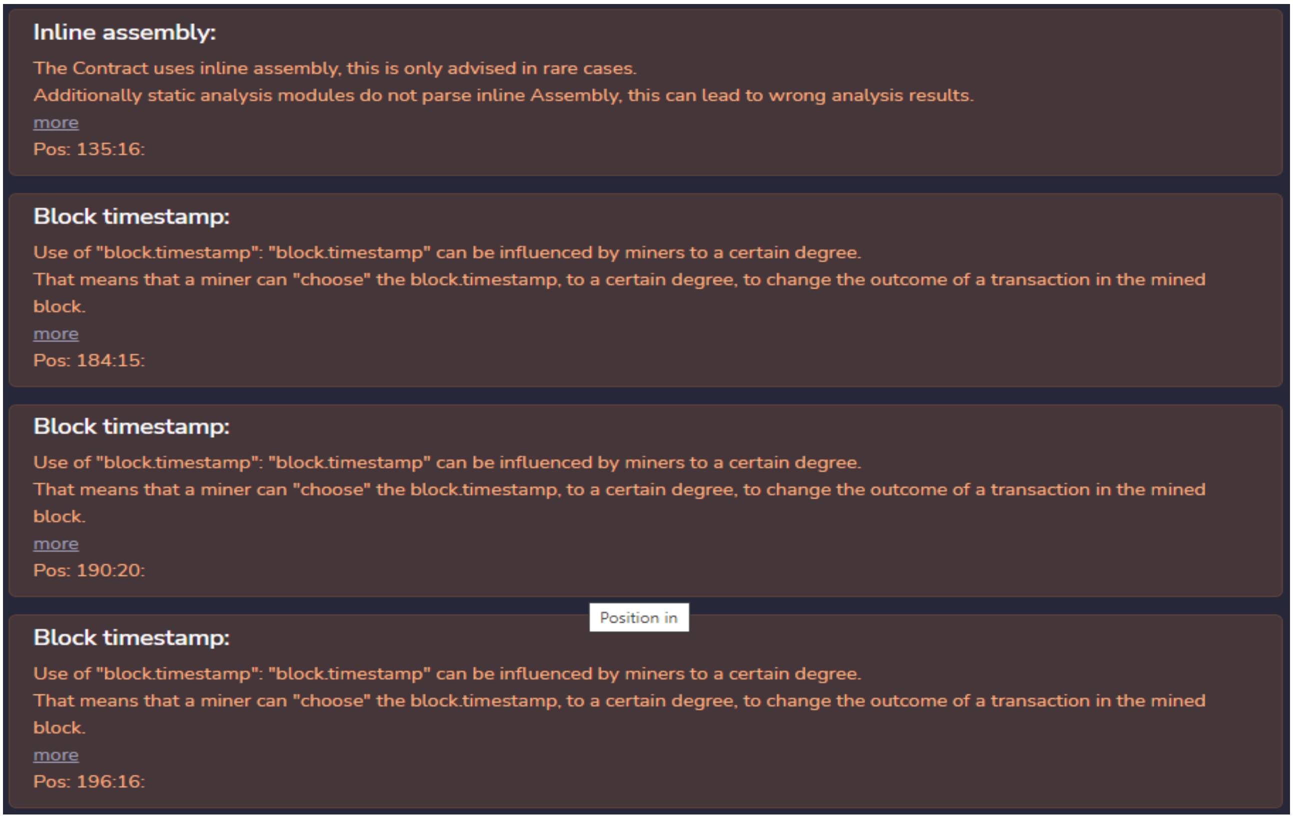
Potential violation of Checks-Effects-Interaction pattern in CoinToken.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 717:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in CoinToken.swapETHForTokens(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 737:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 96:8:



Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 135:16:

Block timestamp:

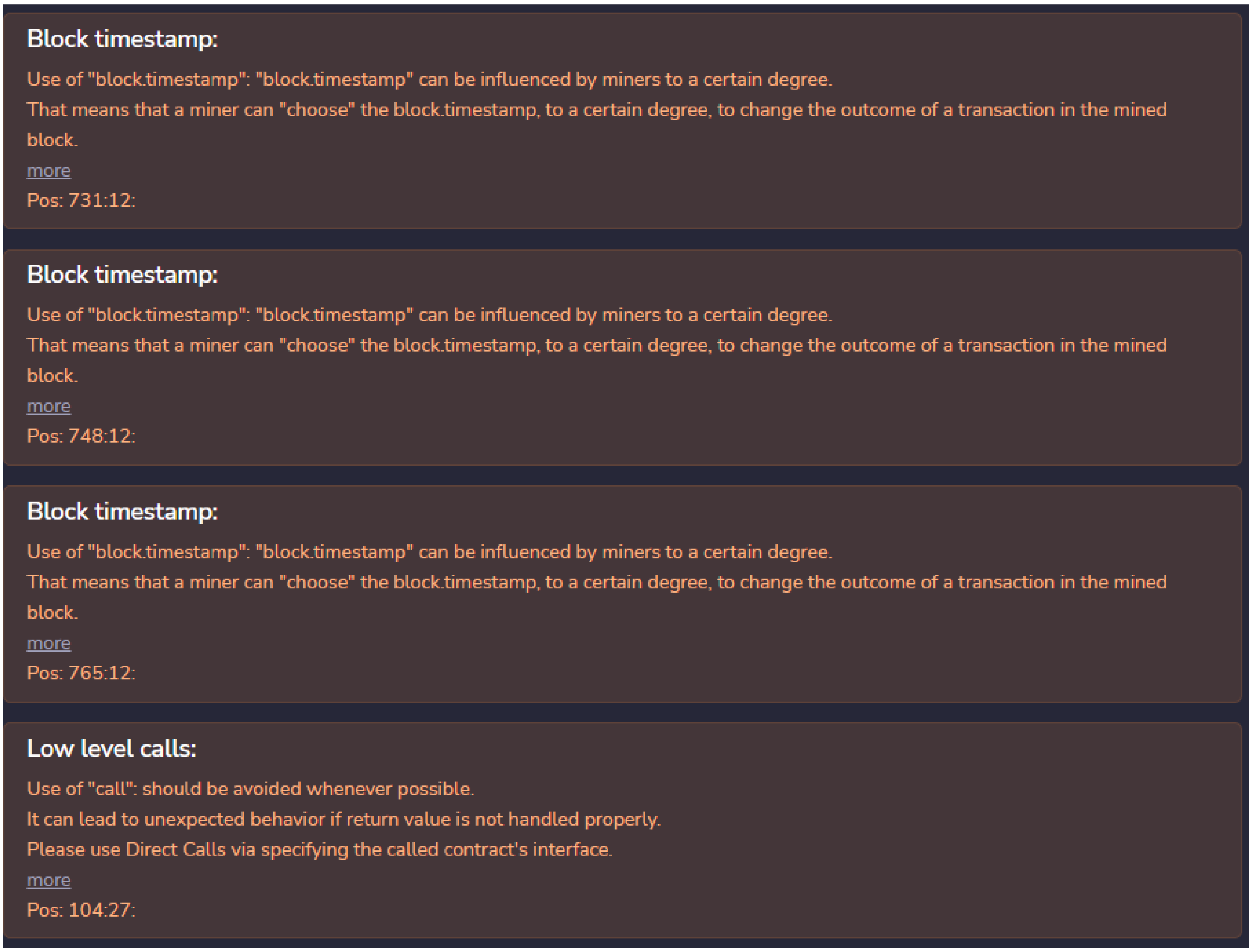
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 184:15:

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: 190:20:

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: 196:16:



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: 731:12:

Block timestamp:

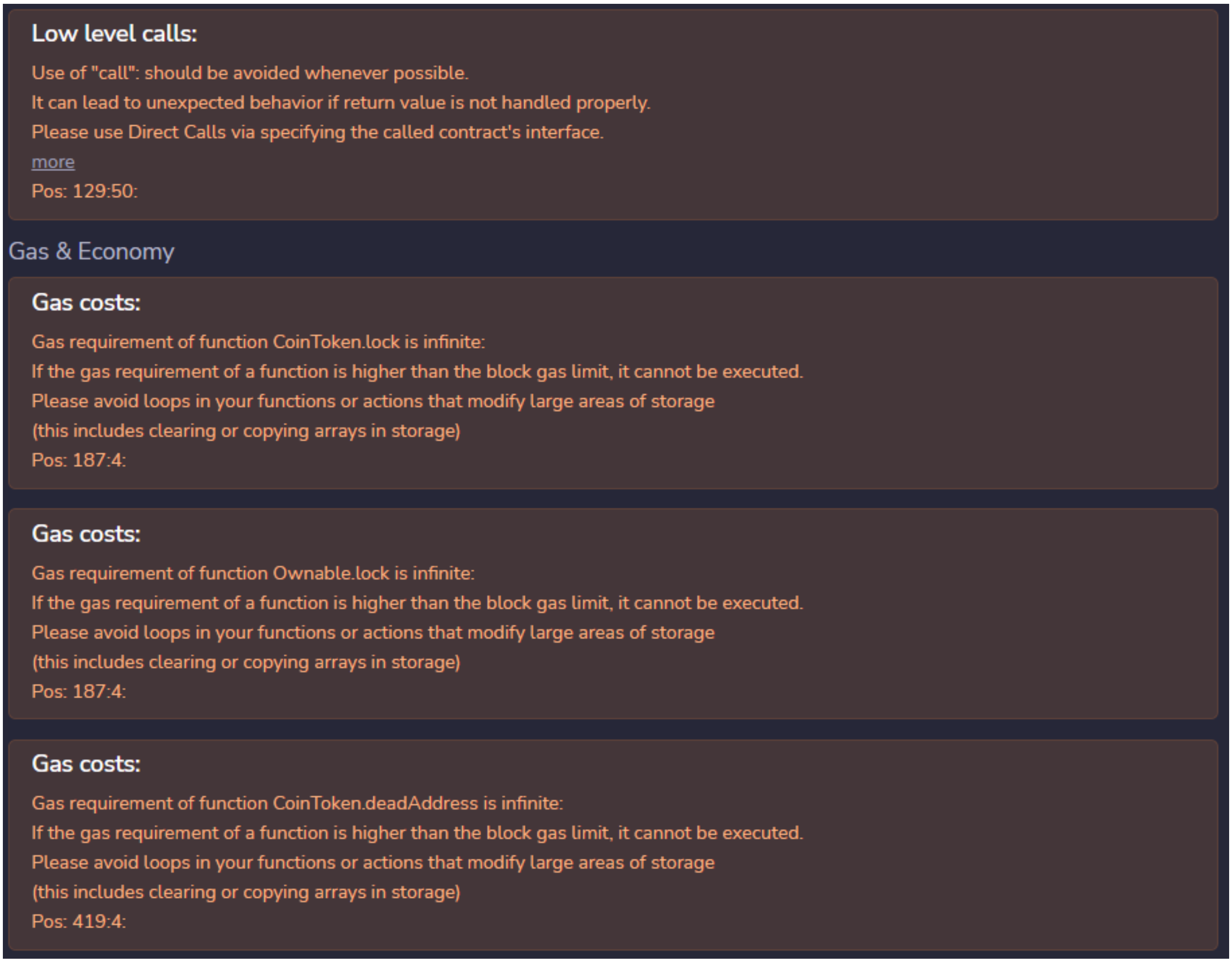
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 748:12:

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: 765:12:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 104:27:



Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 129:50:

Gas & Economy

Gas costs:

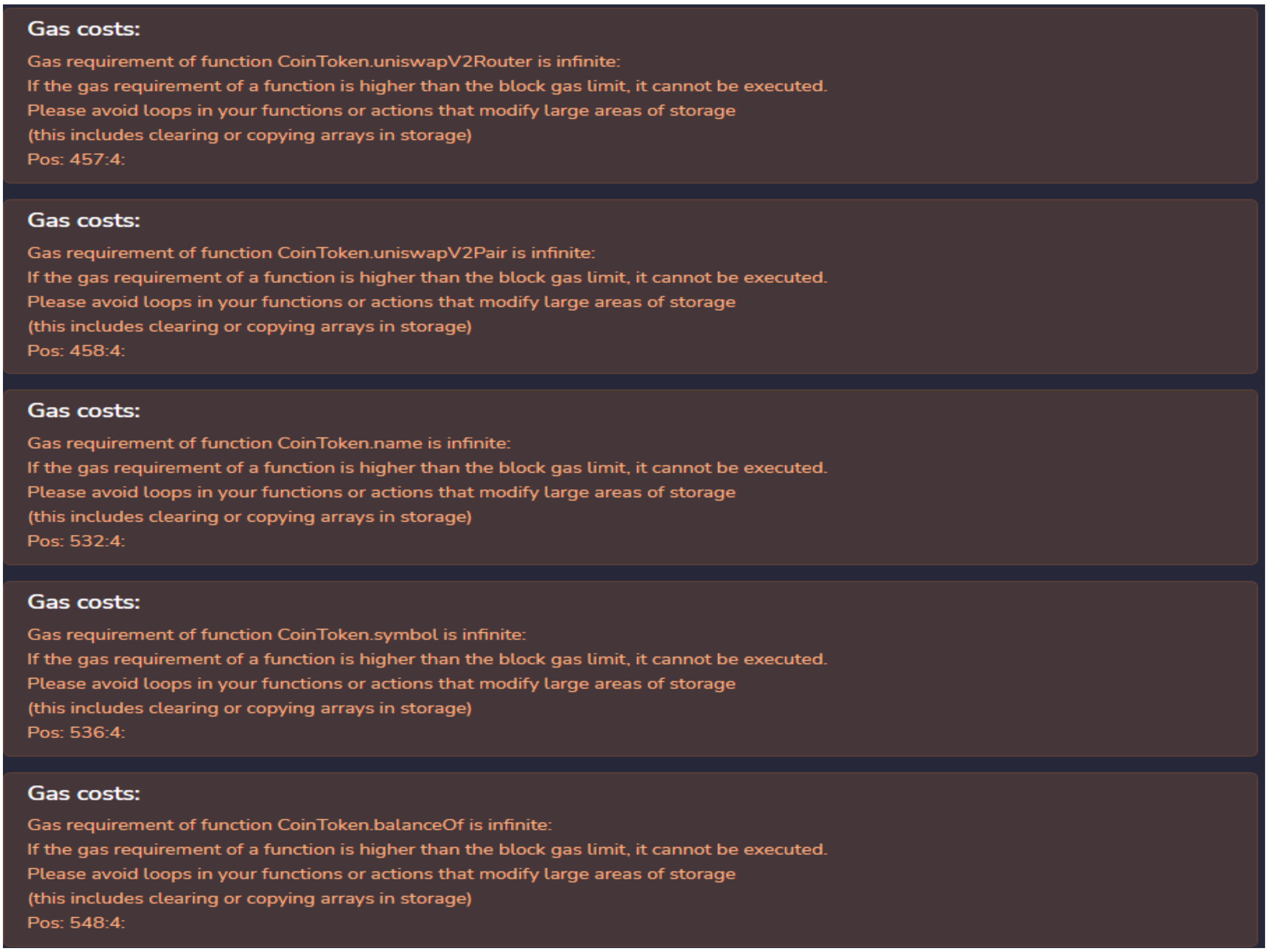
Gas requirement of function CoinToken.lock is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 187:4:

Gas costs:

Gas requirement of function Ownable.lock 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: 187:4:

Gas costs:

Gas requirement of function CoinToken.deadAddress 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: 419:4:



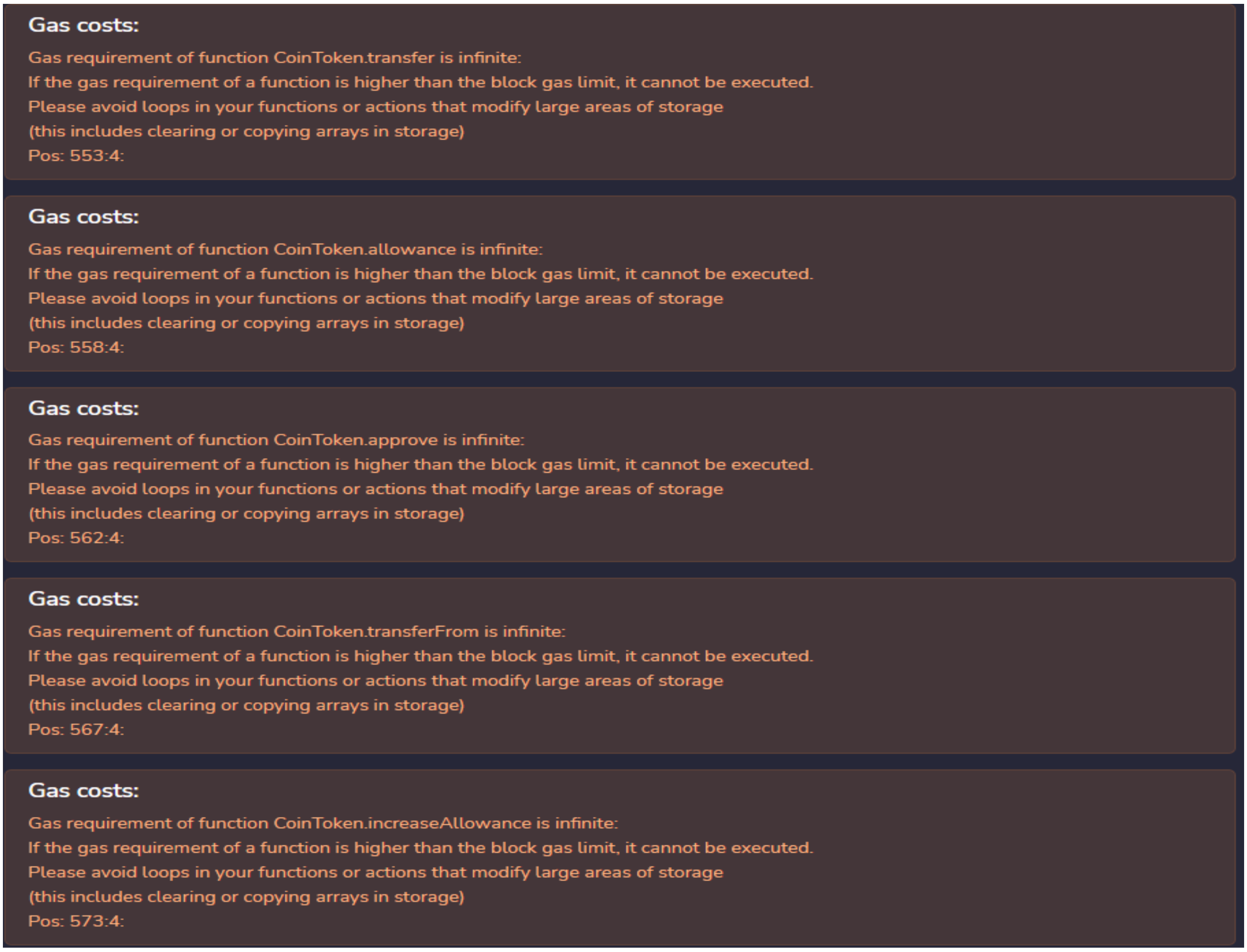
```
Gas costs:
Gas requirement of function CoinToken.uniswapV2Router is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 457:4:

Gas costs:
Gas requirement of function CoinToken.uniswapV2Pair 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: 458:4:

Gas costs:
Gas requirement of function CoinToken.name 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: 532:4:

Gas costs:
Gas requirement of function CoinToken.symbol 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: 536:4:

Gas costs:
Gas requirement of function CoinToken.balanceOf 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: 548:4:
```



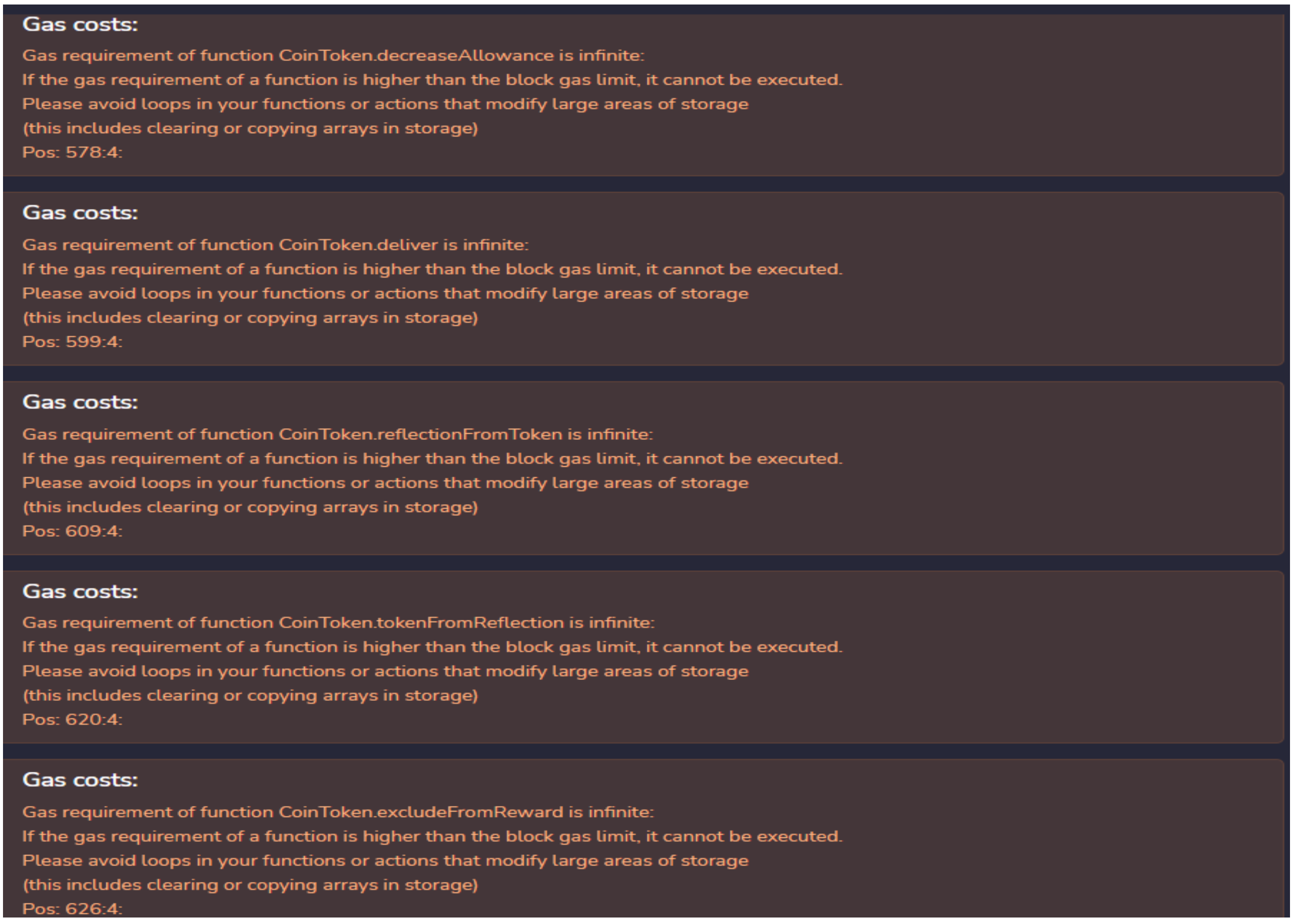
```
Gas costs:
Gas requirement of function CoinToken.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 553:4:

Gas costs:
Gas requirement of function CoinToken.allowance 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: 558:4:

Gas costs:
Gas requirement of function CoinToken.approve 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: 562:4:

Gas costs:
Gas requirement of function CoinToken.transferFrom 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: 567:4:

Gas costs:
Gas requirement of function CoinToken.increaseAllowance 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: 573:4:
```



Gas costs:

Gas requirement of function CoinToken.decreaseAllowance 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: 578:4:

Gas costs:

Gas requirement of function CoinToken.deliver 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: 599:4:

Gas costs:

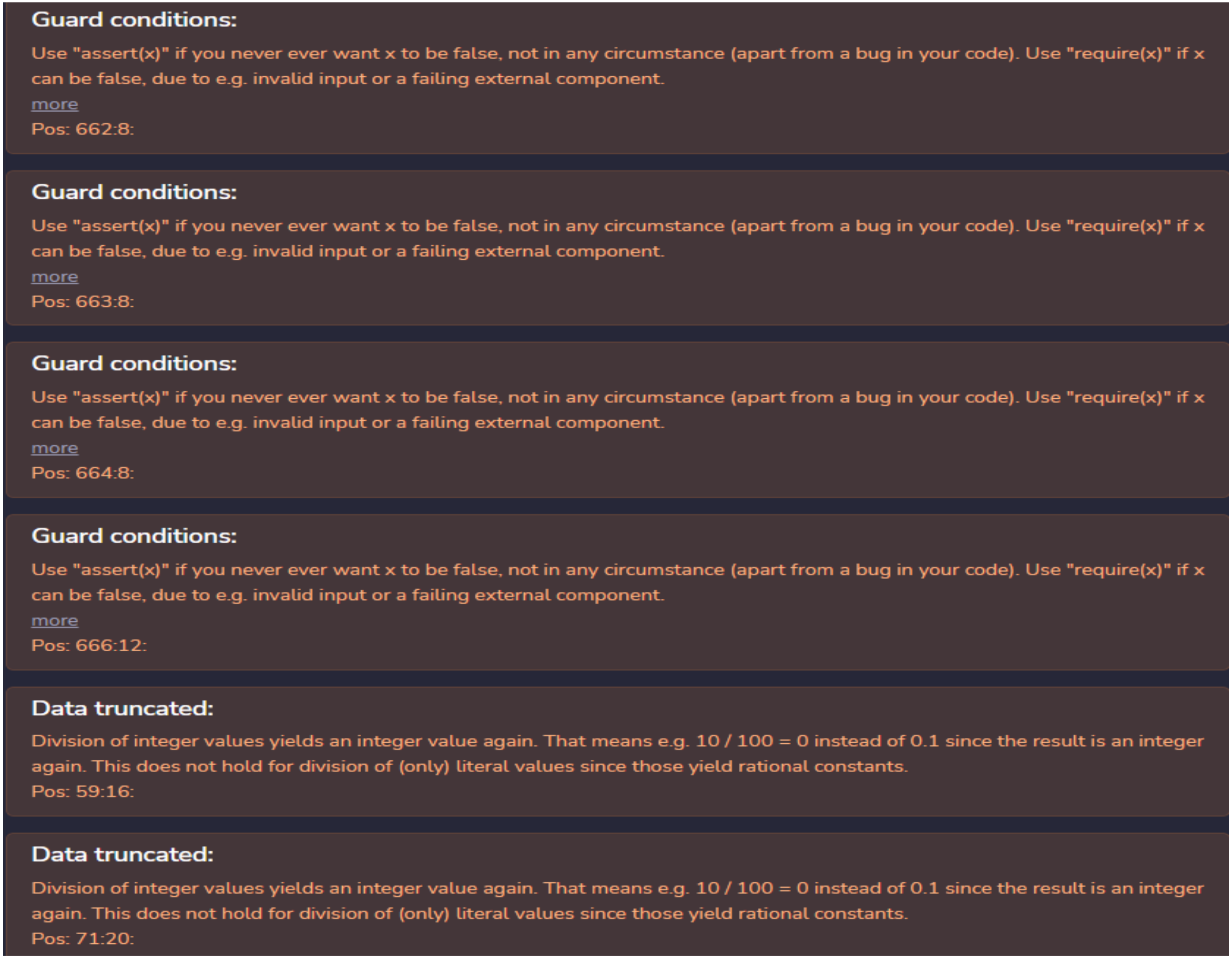
Gas requirement of function CoinToken.reflectionFromToken is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 609:4:

Gas costs:

Gas requirement of function CoinToken.tokenFromReflection 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: 620:4:

Gas costs:

Gas requirement of function CoinToken.excludeFromReward 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: 626:4:



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: 662:8:

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: 663:8:

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: 664:8:

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: 666:12:

Data truncated:

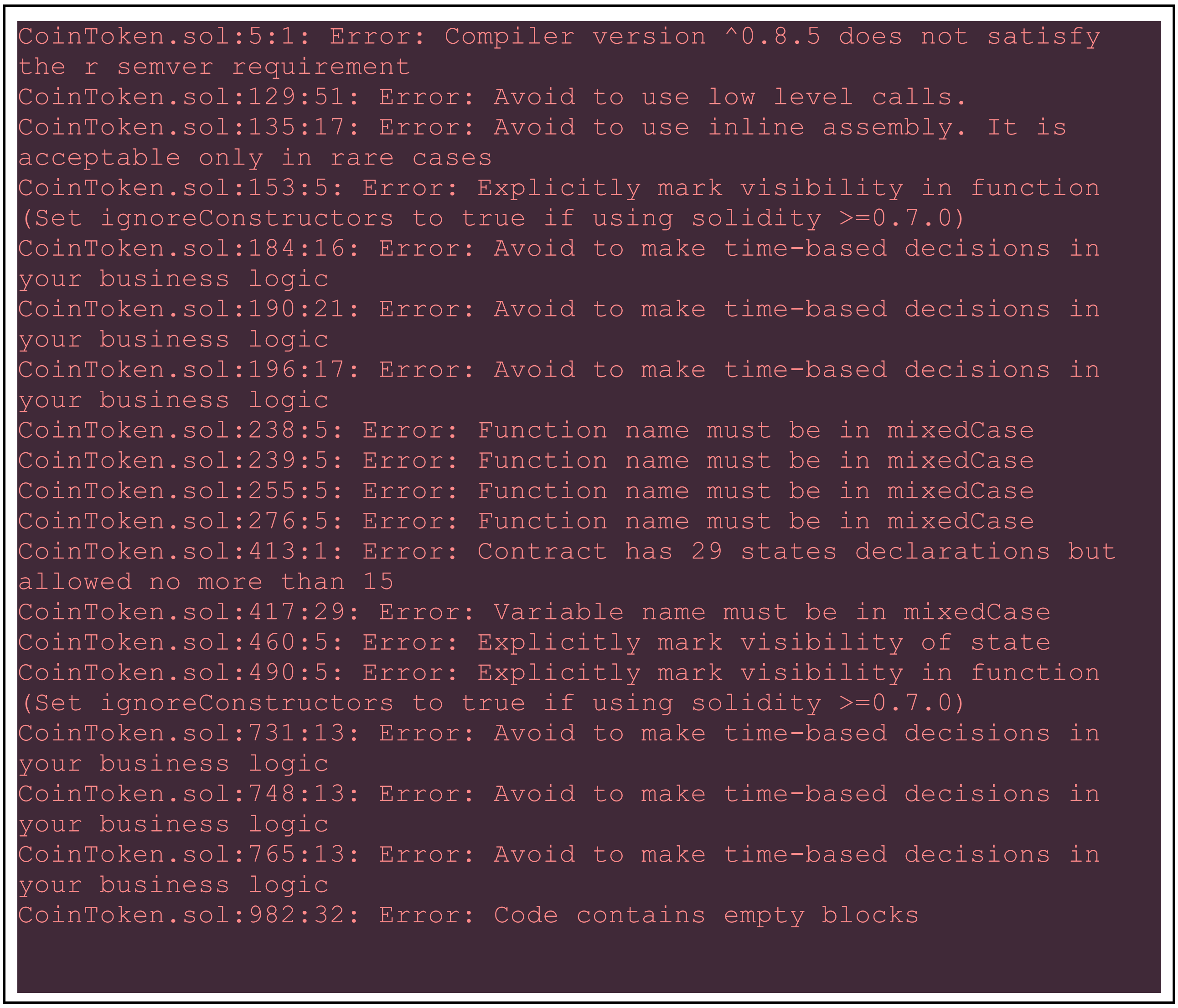
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 59:16:

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: 71:20:

Solhint Linter

CoinToken.sol



```
CoinToken.sol:5:1: Error: Compiler version ^0.8.5 does not satisfy
the r semver requirement
CoinToken.sol:129:51: Error: Avoid to use low level calls.
CoinToken.sol:135:17: Error: Avoid to use inline assembly. It is
acceptable only in rare cases
CoinToken.sol:153:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
CoinToken.sol:184:16: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:190:21: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:196:17: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:238:5: Error: Function name must be in mixedCase
CoinToken.sol:239:5: Error: Function name must be in mixedCase
CoinToken.sol:255:5: Error: Function name must be in mixedCase
CoinToken.sol:276:5: Error: Function name must be in mixedCase
CoinToken.sol:413:1: Error: Contract has 29 states declarations but
allowed no more than 15
CoinToken.sol:417:29: Error: Variable name must be in mixedCase
CoinToken.sol:460:5: Error: Explicitly mark visibility of state
CoinToken.sol:490:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
CoinToken.sol:731:13: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:748:13: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:765:13: Error: Avoid to make time-based decisions in
your business logic
CoinToken.sol:982:32: 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.

EtherAuthority