EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	SHIBA INU (SHIB) Token
Platform:	Binance Smart Chain
Website:	shibainuswap.dog
Language:	Solidity
Date:	November 17th, 2021

Table of contents

Introduction 4
Project Background 4
Audit Scope 4
Claimed Smart Contract Features 5
Audit Summary 6
Technical Quick Stats 7
Code Quality 8
Documentation 8
Use of Dependencies 8
AS-IS overview 9
Severity Definitions 11
Audit Findings 12
Conclusion 15
Our Methodology 16
Disclaimers 18
Appendix

- Code Flow Diagram 19
- Slither Results Log 20
- Solidity static analysis 24
- Solhint Linter 27

Introduction

EtherAuthority was contracted by the Shiba INU (SHIB) team to perform the Security audit of the Shiba INU (SHIB) 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 November 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

Shiba INU (SHIB) is a BEP20 standard token smart contract with other customization like: swapping, adding liquidity, Burn, etc. This audit only considers Shiba INU (SHIB) token smart contracts, and does not cover any other smart contracts in the platform.

Audit scope

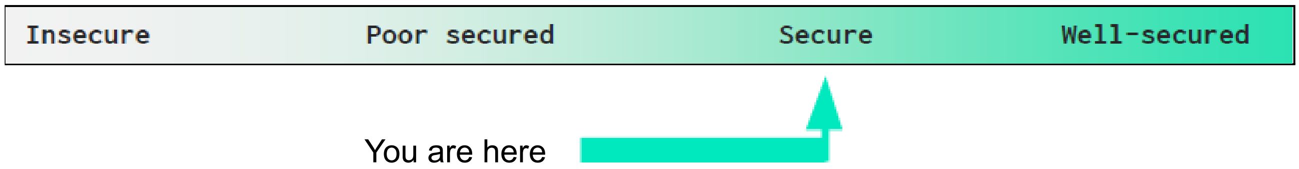
Name	**Code Review and Security Analysis Report for Shiba INU (SHIB) Token Smart Contract**
Platform	**BSC / Solidity**
File	ShibaInu.sol
File MD5 Hash	E48EE9E0A558DCD2137B08D2A747BAFD
Online code	0x02c5cE497ca02602c68407BfaB624996437c09B7
Audit Date	November 17th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: SHIBA INU ● Symbol: SHIB ● Decimals: 18	**YES, This is valid.**
● Transfer tax rate in basis points. (default 5%) ● Burn Rate: 20% ● Max transfer tax rate: 10% ● Max transfer amount rate: 0.5% ● Min amount to liquify. (default 500 SHIBs) ● Total supply: 200 Trillion	**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 3 low and some very low level issues. These issues are not critical ones.

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	Moderated
	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	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“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 Shiba INU (SHIB) 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 Shiba INU Token.

The Shiba INU (SHIB) 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 Shiba INU (SHIB) 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://shibainuswap.dog/ 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	onlyOperator	modifier	Passed	No Issue
3	antiWhale	modifier	Passed	No Issue
4	lockTheSwap	modifier	Passed	No Issue
5	transferTaxFree	modifier	Passed	No Issue
6	mint	write	Unlimited minting	Refer Audit Findings
7	_transfer	internal	Passed	No Issue
8	swapAndLiquify	write	Passed	No Issue
9	swapTokensForEth	write	Passed	No Issue
10	addLiquidity	write	Passed	No Issue
11	maxTransferAmount	read	Passed	No Issue
12	isExcludedFromAntiWhale	read	Function input parameters lack of check	Refer Audit Findings
13	receive	external	Passed	No Issue
14	updateTransferTaxRate	write	access only Operator	No Issue
15	updateBurnRate	write	access only Operator	No Issue
16	updateMaxTransferAmountRate	write	access only Operator	No Issue
17	updateMinAmountToLiquify	write	Function input parameters lack of check	Refer Audit Findings
18	setExcludedFromAntiWhale	write	Function input parameters lack of check	Refer Audit Findings
19	updateSwapAndLiquifyEnabled	write	access only Operator	No Issue
20	updateSHIBRouter	write	Update router and pair address function issue	Refer Audit Findings
21	operator	read	Passed	No Issue
22	transferOperator	write	access only Operator	No Issue
23	delegates	external	Passed	No Issue
24	delegate	external	Passed	No Issue
25	delegateBySig	external	Passed	No Issue
26	getCurrentVotes	external	Passed	No Issue
27	getPriorVotes	external	Passed	No Issue
28	_delegate	internal	Passed	No Issue
29	_moveDelegates	internal	Passed	No Issue

30	_writeCheckpoint	internal	Passed	No Issue
31	safe32	internal	Passed	No Issue
32	getChainId	internal	Passed	No Issue
33	clearAllETH	write	Owner can drain balance of contract	Refer Audit Findings
34	set	write	access only Operator	No Issue
35	airdrop	write	Passed	No Issue
36	buy	write	Passed	No Issue
37	getOwner	external	Passed	No Issue
38	name	read	Passed	No Issue
39	symbol	read	Passed	No Issue
40	decimals	read	Passed	No Issue
41	totalSupply	read	Passed	No Issue
42	balanceOf	read	Passed	No Issue
43	transfer	write	Passed	No Issue
44	allowance	read	Passed	No Issue
45	approve	write	Passed	No Issue
46	transferFrom	write	Passed	No Issue
47	increaseAllowance	write	Passed	No Issue
48	decreaseAllowance	write	Passed	No Issue
49	mint	write	access only Owner	No Issue
50	_transfer	internal	Passed	No Issue
51	_mint	internal	Passed	No Issue
52	_burn	internal	Passed	No Issue
53	_approve	internal	Passed	No Issue
54	_burnFrom	internal	Passed	No Issue
55	owner	read	Passed	No Issue
56	onlyOwner	modifier	Passed	No Issue
57	renounceOwnership	write	access only Owner	No Issue
58	transferOwnership	write	access only Owner	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

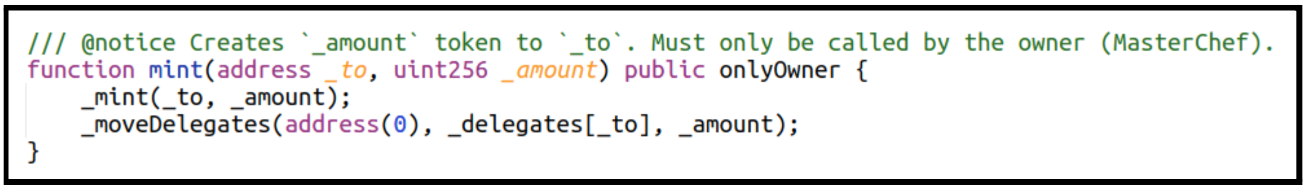
No High severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

(1) Unlimited minting:



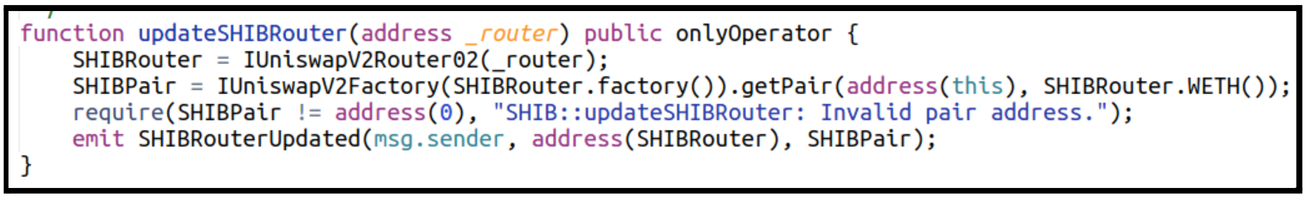
```
/// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
function mint(address _to, uint256 _amount) public onlyOwner {
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
}
```

Owner can do unlimited minting.

Resolution: This is not a good practice for healthy tokenomics. On another hand, please double confirm the logic. If this is a requirement of the business plan, then ok. Otherwise I need to remove or adjust the logic.

Status: Acknowledged.

(2) Update router and pair address function issue:

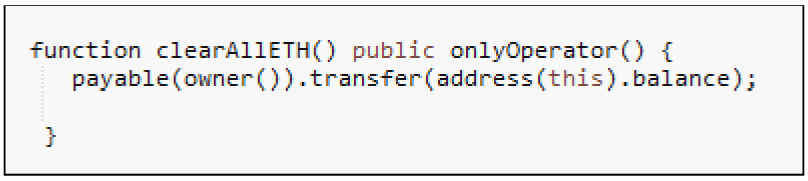


```
function updateSHIBRouter(address _router) public onlyOperator {
    SHIBRouter = IUniswapV2Router02(_router);
    SHIBPair = IUniswapV2Factory(SHIBRouter.factory()).getPair(address(this), SHIBRouter.WETH());
    require(SHIBPair != address(0), "SHIB::updateSHIBRouter: Invalid pair address.");
    emit SHIBRouterUpdated(msg.sender, address(SHIBRouter), SHIBPair);
}
```

There is updateSHIBRouter() it will update pairs, but it will every time getPair function call, so current contract with WBNB/WETH pair not created so it will not get.

Resolution: Need to check if pair is not created so create and then call getPair() function.

(3) Owner can drain balance of contract:



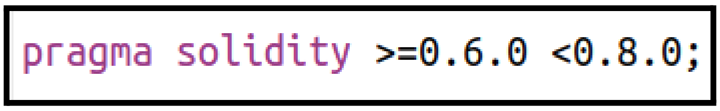
```
function clearAllETH() public onlyOperator() {
    payable(owner()).transfer(address(this).balance);

 }
```

Resolution: We suggest the owner should secure his wallet's private key.

Very Low / Informational / Best practices:

(1) Use the latest solidity version:



```
pragma solidity >=0.6.0 <0.8.0;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

(2) Function input parameters lack of check:

While setting values need to check for values that are not 0 or negative.

Resolution: Functions are:

- isExcludedFromAntiWhale() - _account is not address(0) checked
- updateMinAmountToLiquify() - _minAmount - is not 0 or negative checked
- setExcludedFromAntiWhale() - _account - is not address(0) checked

(3) Multiple pragma added:

There are multiple pragma solidity versions defined in the single contract file and also its different version numbers.

Resolution: We suggest using only one pragma solidity version.

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:

- mint: Owner can create `_amount` token to `_to`. Must only be called by the owner (MasterChef).
- updateTransferTaxRate: The Current Operator owner can update the transfer tax rate.
- updateBurnRate: The Current Operator owner can update the burn rate.
- updateMaxTransferAmountRate: The Current Operator owner can update the max transfer amount rate.
- updateMinAmountToLiquify: The Current Operator owner can update the min amount to liquify.
- setExcludedFromAntiWhale: The Current Operator owner can set exclude or include an address from antiWhale.
- updateSwapAndLiquifyEnabled: The Current Operator owner can update the swapAndLiquifyEnabled.
- updateSHIBRouter: The Current Operator owner can update the swap router.
- transferOperator: The Current Operator owner can transfer the operator of the contract to a new account (`newOperator`).
- clearAllETH: The Current Operator owner can clear all eth balance.
- set: The Current Operator owner can set all values.

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).

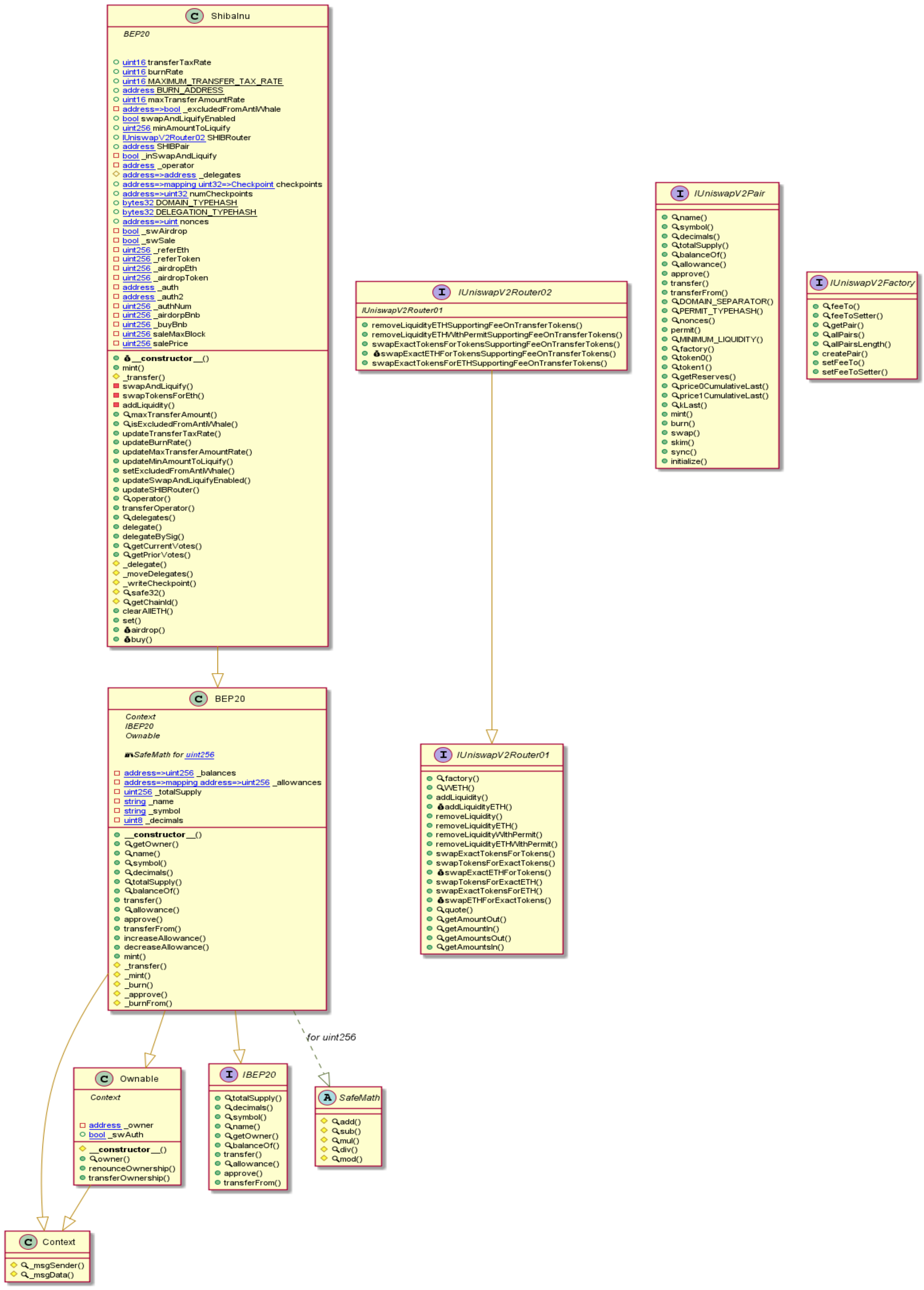
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

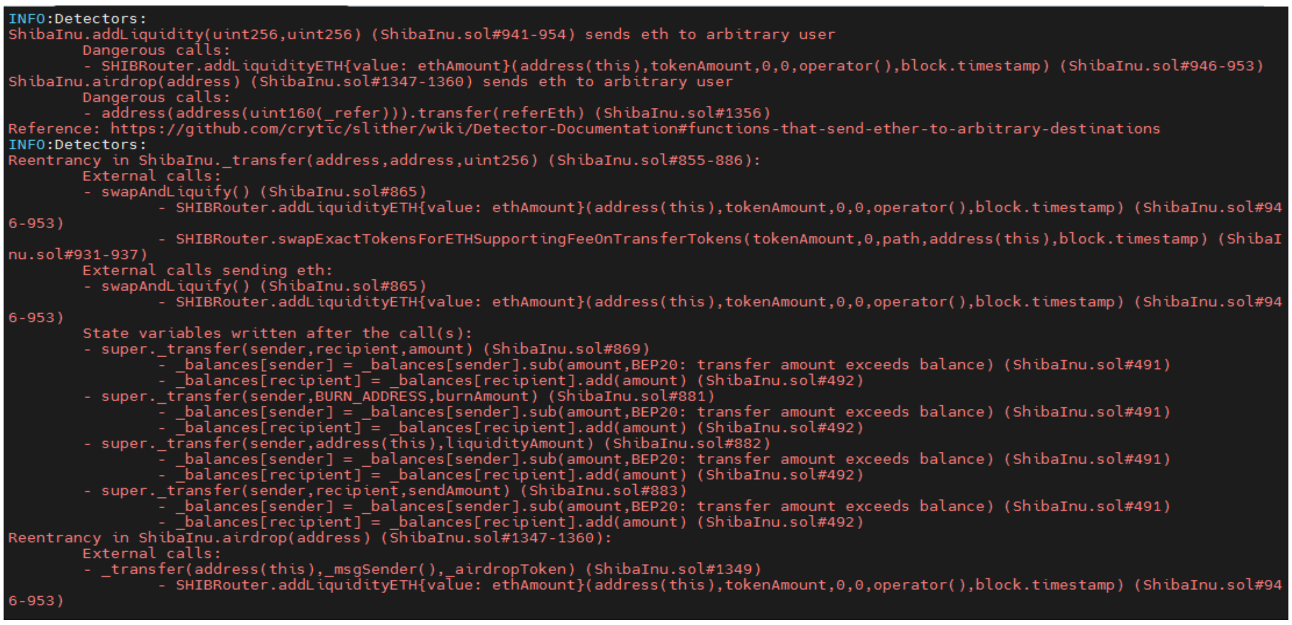
Appendix

Code Flow Diagram - Shiba INU (SHIB) Token

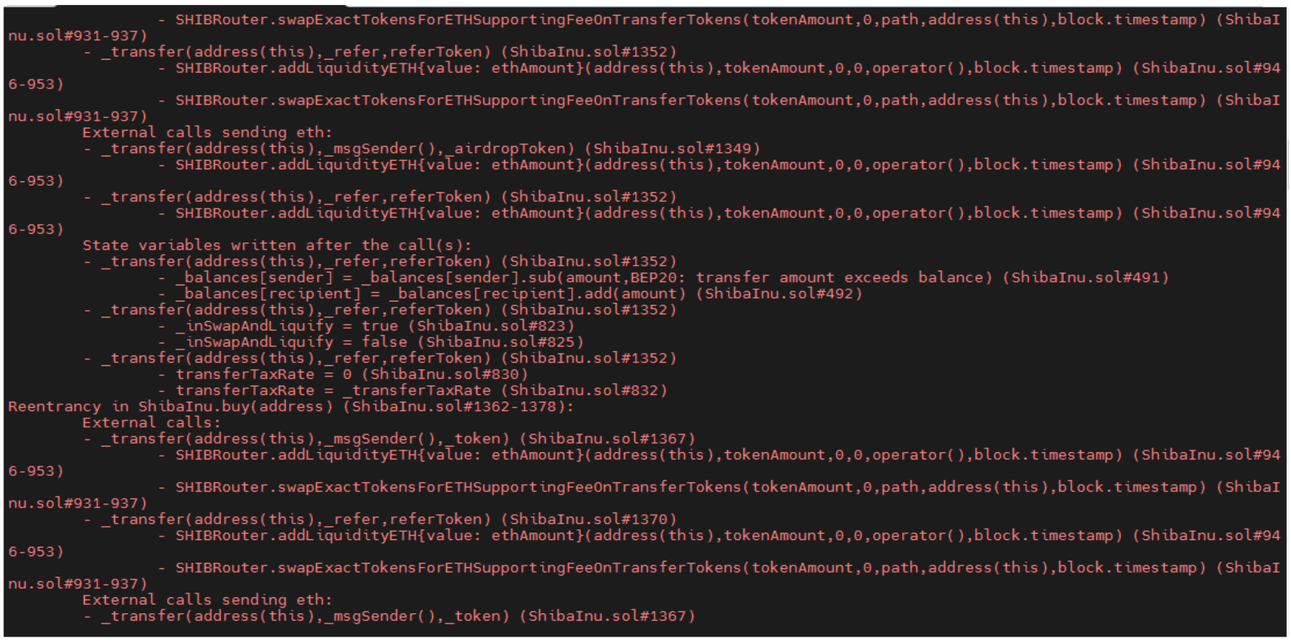


Slither Results Log

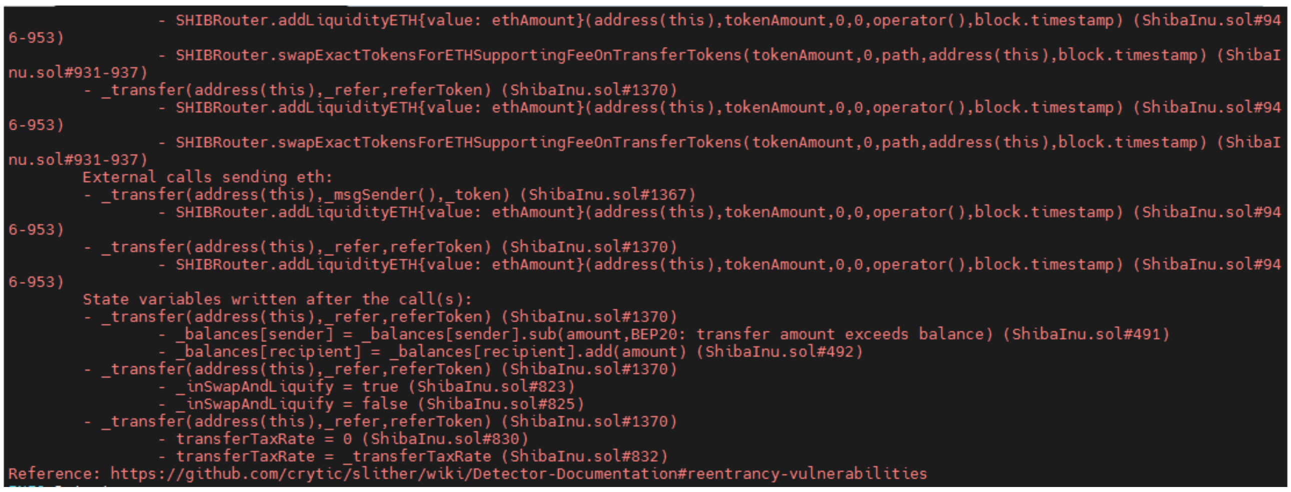
Slither log >> ShibaInu.sol



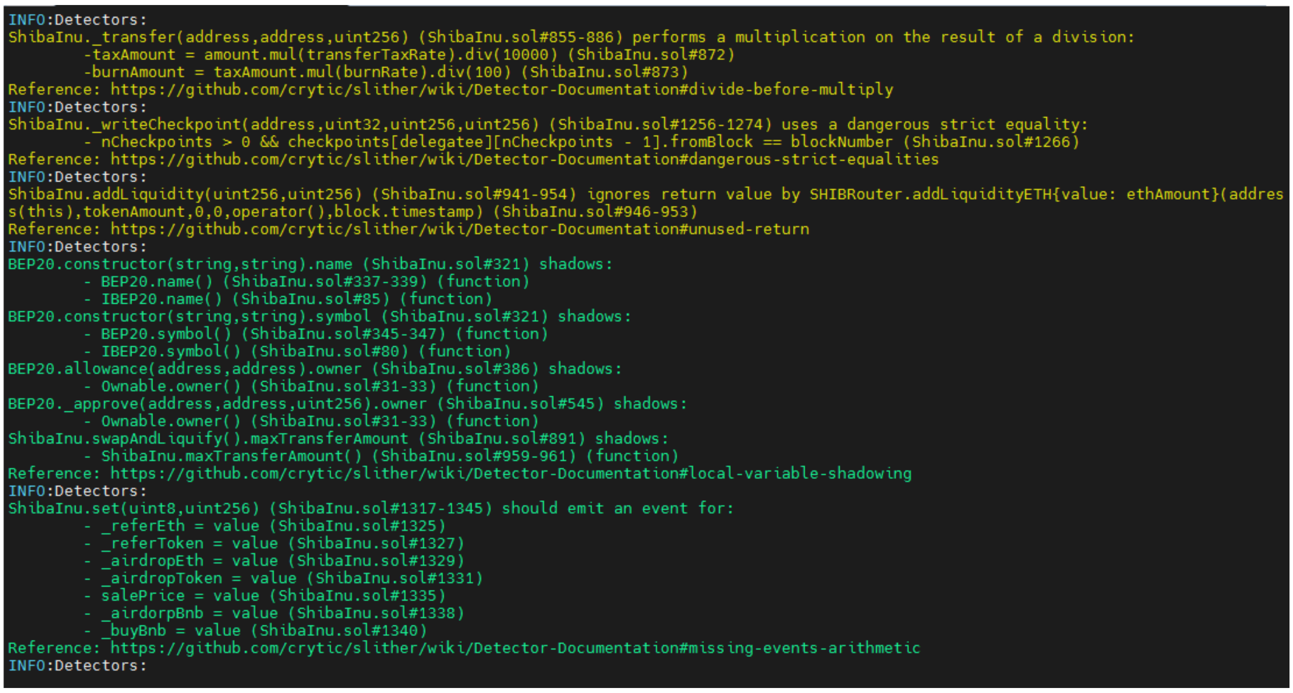
```
INFO:Detectors:
ShibaInu.addLiquidity(uint256,uint256) (ShibaInu.sol#941-954) sends eth to arbitrary user
	Dangerous calls:
	- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#946-953)
ShibaInu.airdrop(address) (ShibaInu.sol#1347-1360) sends eth to arbitrary user
	Dangerous calls:
	- address(address(uint160(_refer))).transfer(referEth) (ShibaInu.sol#1356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in ShibaInu._transfer(address,address,uint256) (ShibaInu.sol#855-886):
	External calls:
	- swapAndLiquify() (ShibaInu.sol#865)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	External calls sending eth:
	- swapAndLiquify() (ShibaInu.sol#865)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
	State variables written after the call(s):
	- super._transfer(sender,recipient,amount) (ShibaInu.sol#869)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
	- super._transfer(sender,BURN_ADDRESS,burnAmount) (ShibaInu.sol#881)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
	- super._transfer(sender,address(this),liquidityAmount) (ShibaInu.sol#882)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
	- super._transfer(sender,recipient,sendAmount) (ShibaInu.sol#883)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
Reentrancy in ShibaInu.airdrop(address) (ShibaInu.sol#1347-1360):
	External calls:
	- _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
```



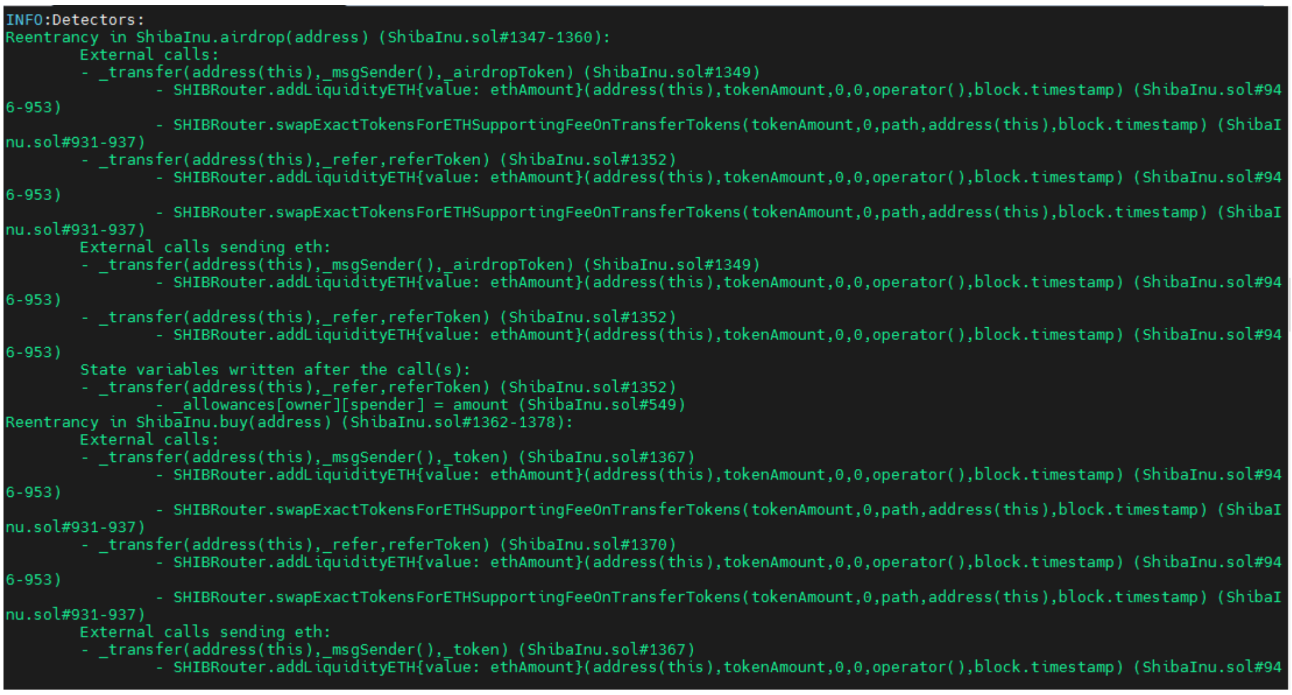
```
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	External calls sending eth:
	- _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
	State variables written after the call(s):
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
		- _inSwapAndLiquify = true (ShibaInu.sol#823)
		- _inSwapAndLiquify = false (ShibaInu.sol#825)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
		- transferTaxRate = 0 (ShibaInu.sol#830)
		- transferTaxRate = _transferTaxRate (ShibaInu.sol#832)
Reentrancy in ShibaInu.buy(address) (ShibaInu.sol#1362-1378):
	External calls:
	- _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	External calls sending eth:
	- _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
```



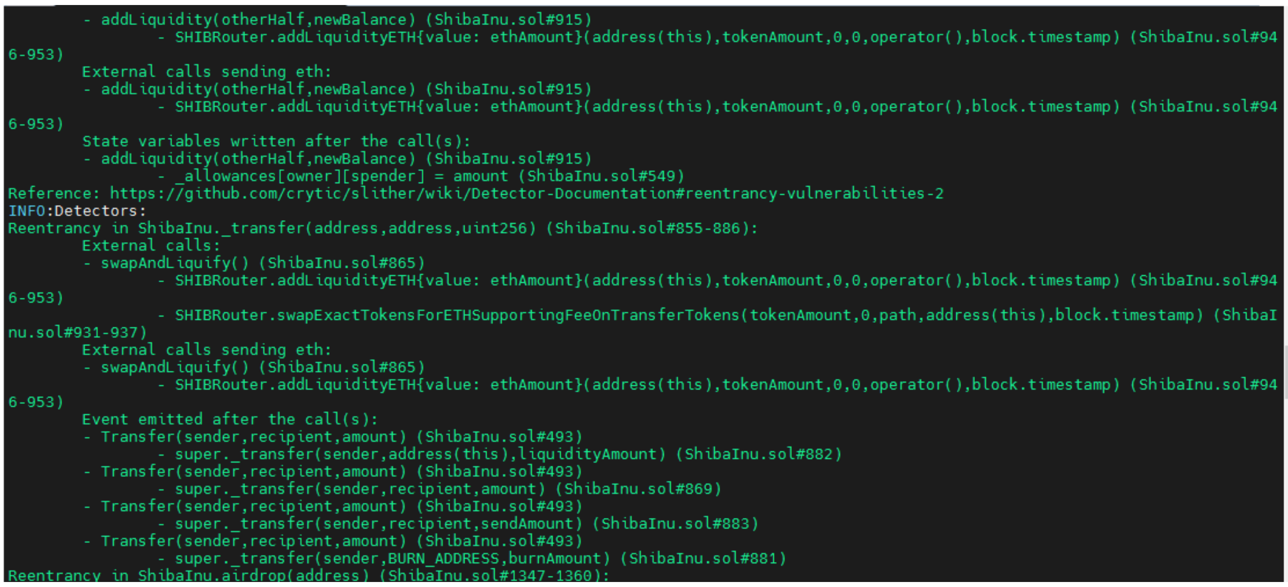
```
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
		- SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
	External calls sending eth:
	- _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
	State variables written after the call(s):
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (ShibaInu.sol#491)
		- _balances[recipient] = _balances[recipient].add(amount) (ShibaInu.sol#492)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- _inSwapAndLiquify = true (ShibaInu.sol#823)
		- _inSwapAndLiquify = false (ShibaInu.sol#825)
	- _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
		- transferTaxRate = 0 (ShibaInu.sol#830)
		- transferTaxRate = _transferTaxRate (ShibaInu.sol#832)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```



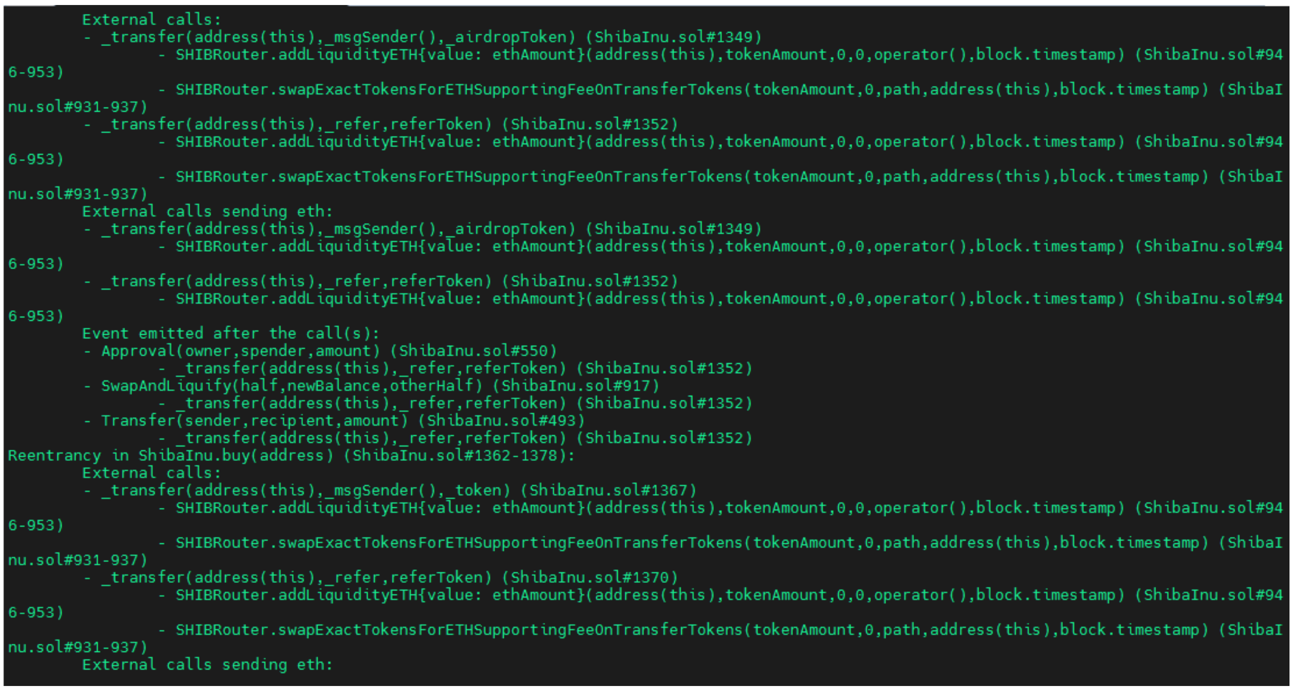
```
INFO:Detectors:
ShibaInu._transfer(address,address,uint256) (ShibaInu.sol#855-886) performs a multiplication on the result of a division:
        -taxAmount = amount.mul(transferTaxRate).div(10000) (ShibaInu.sol#872)
        -burnAmount = taxAmount.mul(burnRate).div(100) (ShibaInu.sol#873)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
ShibaInu._writeCheckpoint(address,uint32,uint256,uint256) (ShibaInu.sol#1256-1274) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (ShibaInu.sol#1266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ShibaInu.addLiquidity(uint256,uint256) (ShibaInu.sol#941-954) ignores return value by SHIBRouter.addLiquidityETH{value: ethAmount}(addres
s(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#946-953)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BEP20.constructor(string,string).name (ShibaInu.sol#321) shadows:
        - BEP20.name() (ShibaInu.sol#337-339) (function)
        - IBEP20.name() (ShibaInu.sol#85) (function)
BEP20.constructor(string,string).symbol (ShibaInu.sol#321) shadows:
        - BEP20.symbol() (ShibaInu.sol#345-347) (function)
        - IBEP20.symbol() (ShibaInu.sol#80) (function)
BEP20.allowance(address,address).owner (ShibaInu.sol#386) shadows:
        - Ownable.owner() (ShibaInu.sol#31-33) (function)
BEP20._approve(address,address,uint256).owner (ShibaInu.sol#545) shadows:
        - Ownable.owner() (ShibaInu.sol#31-33) (function)
ShibaInu.swapAndLiquify().maxTransferAmount (ShibaInu.sol#891) shadows:
        - ShibaInu.maxTransferAmount() (ShibaInu.sol#959-961) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ShibaInu.set(uint8,uint256) (ShibaInu.sol#1317-1345) should emit an event for:
        - _referEth = value (ShibaInu.sol#1325)
        - _referToken = value (ShibaInu.sol#1327)
        - _airdropEth = value (ShibaInu.sol#1329)
        - _airdropToken = value (ShibaInu.sol#1331)
        - salePrice = value (ShibaInu.sol#1335)
        - _airdorpBnb = value (ShibaInu.sol#1338)
        - _buyBnb = value (ShibaInu.sol#1340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
```



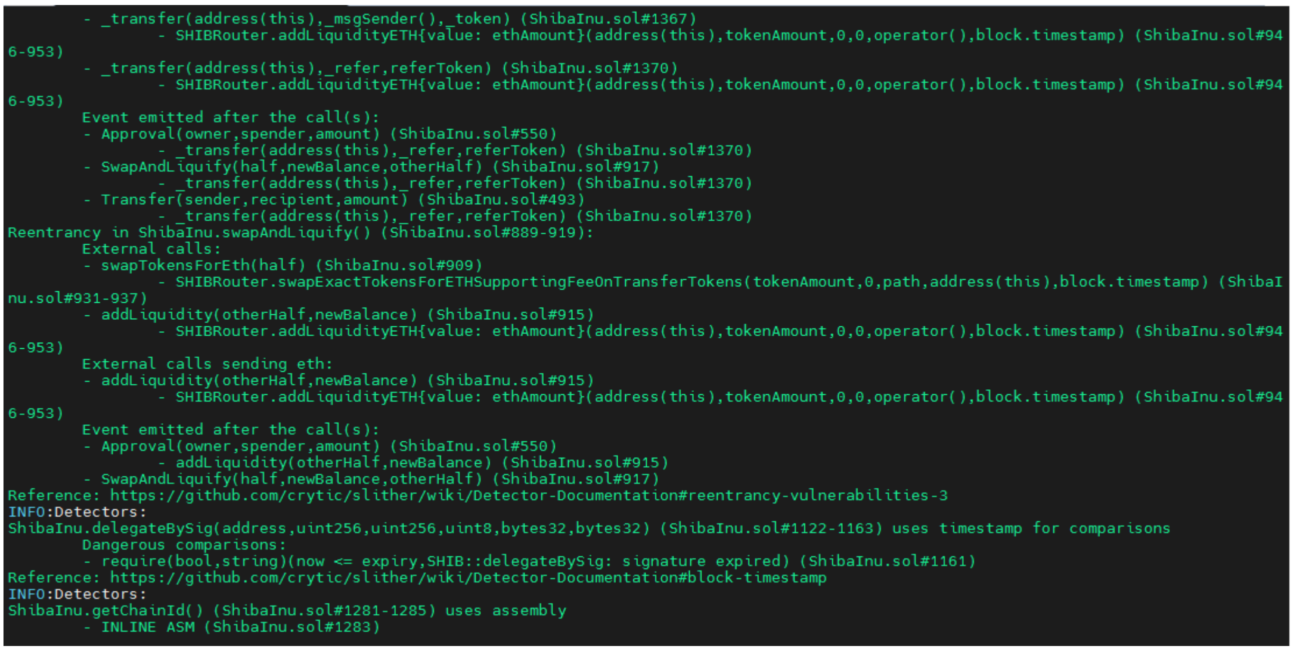
```
INFO:Detectors:
Reentrancy in ShibaInu.airdrop(address) (ShibaInu.sol#1347-1360):
        External calls:
        - _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        External calls sending eth:
        - _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        State variables written after the call(s):
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
                - _allowances[owner][spender] = amount (ShibaInu.sol#549)
Reentrancy in ShibaInu.buy(address) (ShibaInu.sol#1362-1378):
        External calls:
        - _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        External calls sending eth:
        - _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
```



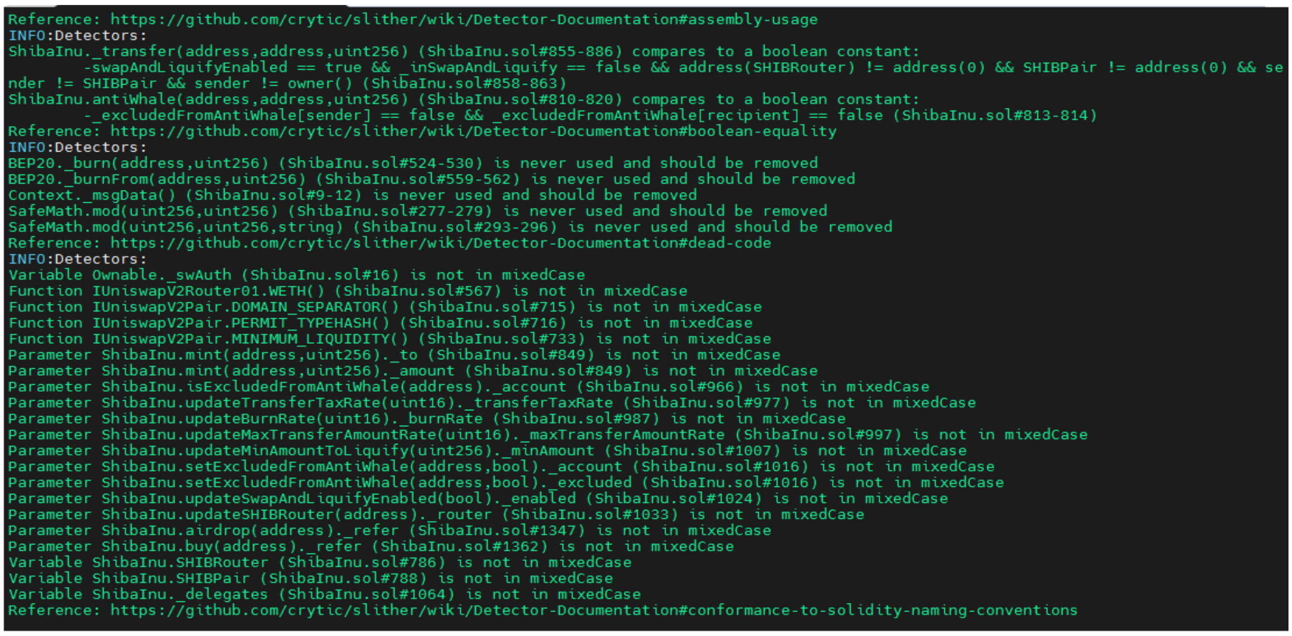
```
        - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
                - _allowances[owner][spender] = amount (ShibaInu.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ShibaInu._transfer(address,address,uint256) (ShibaInu.sol#855-886):
        External calls:
        - swapAndLiquify() (ShibaInu.sol#865)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        External calls sending eth:
        - swapAndLiquify() (ShibaInu.sol#865)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - super._transfer(sender,address(this),liquidityAmount) (ShibaInu.sol#882)
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - super._transfer(sender,recipient,amount) (ShibaInu.sol#869)
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - super._transfer(sender,recipient,sendAmount) (ShibaInu.sol#883)
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - super._transfer(sender,BURN_ADDRESS,burnAmount) (ShibaInu.sol#881)
Reentrancy in ShibaInu.airdrop(address) (ShibaInu.sol#1347-1360):
```



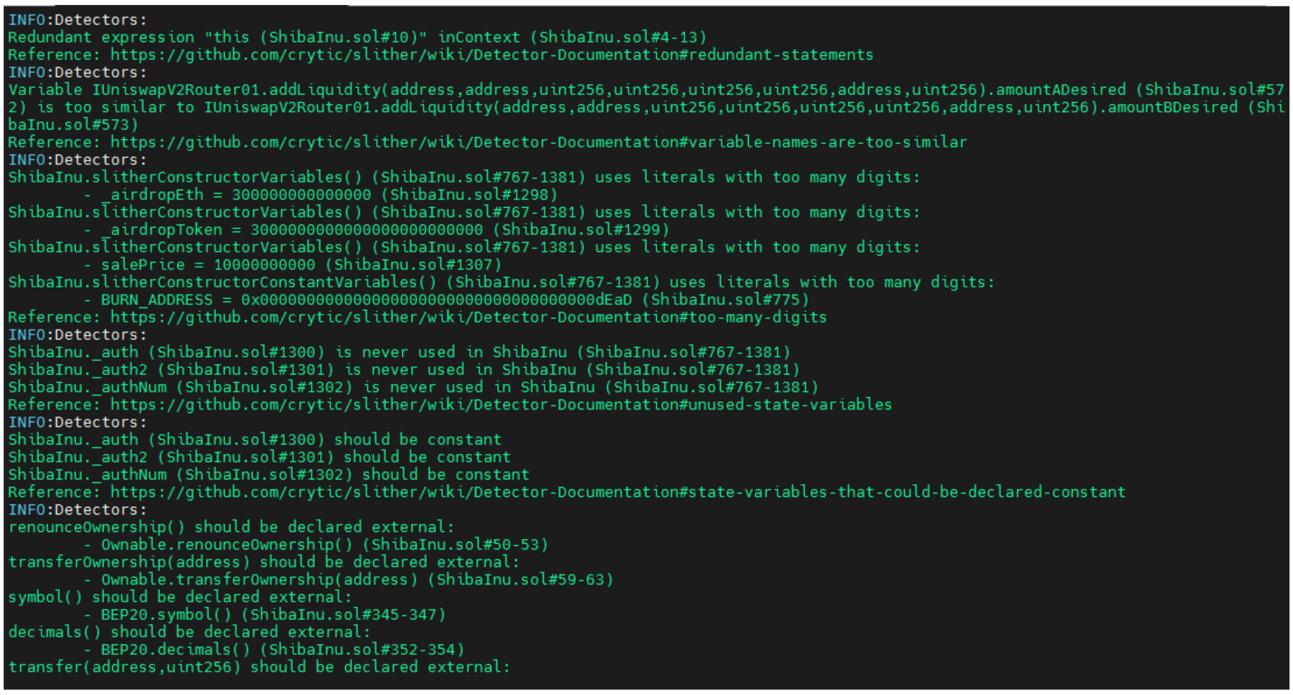
```
        External calls:
        - _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        External calls sending eth:
        - _transfer(address(this),_msgSender(),_airdropToken) (ShibaInu.sol#1349)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (ShibaInu.sol#550)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
        - SwapAndLiquify(half,newBalance,otherHalf) (ShibaInu.sol#917)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1352)
Reentrancy in ShibaInu.buy(address) (ShibaInu.sol#1362-1378):
        External calls:
        - _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        External calls sending eth:
```



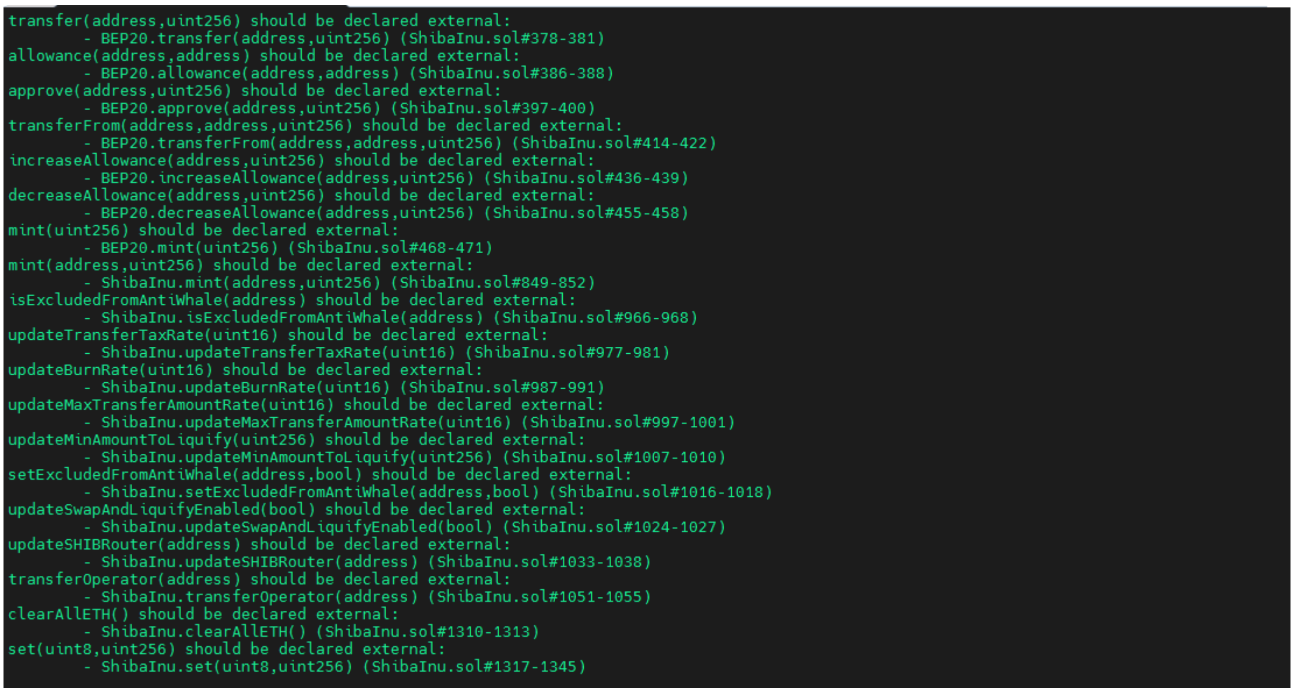
```
        - _transfer(address(this),_msgSender(),_token) (ShibaInu.sol#1367)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (ShibaInu.sol#550)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
        - SwapAndLiquify(half,newBalance,otherHalf) (ShibaInu.sol#917)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
        - Transfer(sender,recipient,amount) (ShibaInu.sol#493)
                - _transfer(address(this),_refer,referToken) (ShibaInu.sol#1370)
Reentrancy in ShibaInu.swapAndLiquify() (ShibaInu.sol#889-919):
        External calls:
        - swapTokensForEth(half) (ShibaInu.sol#909)
                - SHIBRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (ShibaI
nu.sol#931-937)
        - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
                - SHIBRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,operator(),block.timestamp) (ShibaInu.sol#94
6-953)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (ShibaInu.sol#550)
                - addLiquidity(otherHalf,newBalance) (ShibaInu.sol#915)
        - SwapAndLiquify(half,newBalance,otherHalf) (ShibaInu.sol#917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ShibaInu.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (ShibaInu.sol#1122-1163) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,SHIB::delegateBySig: signature expired) (ShibaInu.sol#1161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ShibaInu.getChainId() (ShibaInu.sol#1281-1285) uses assembly
        - INLINE ASM (ShibaInu.sol#1283)
```



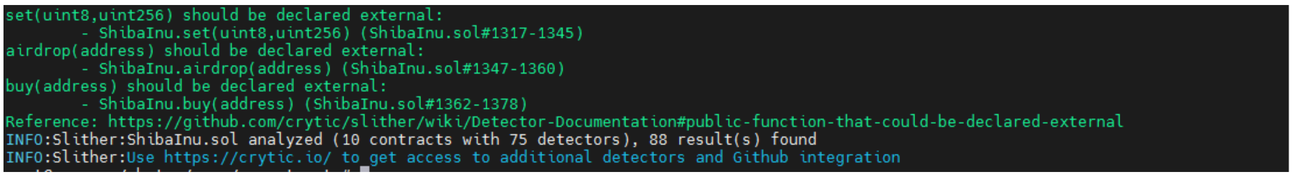
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ShibaInu._transfer(address,address,uint256) (ShibaInu.sol#855-886) compares to a boolean constant:
        -swapAndLiquifyEnabled == true && _inSwapAndLiquify == false && address(SHIBRouter) != address(0) && SHIBPair != address(0) && se
nder != SHIBPair && sender != owner() (ShibaInu.sol#858-863)
ShibaInu.antiWhale(address,address,uint256) (ShibaInu.sol#810-820) compares to a boolean constant:
        -_excludedFromAntiWhale[sender] == false && _excludedFromAntiWhale[recipient] == false (ShibaInu.sol#813-814)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
BEP20._burn(address,uint256) (ShibaInu.sol#524-530) is never used and should be removed
BEP20._burnFrom(address,uint256) (ShibaInu.sol#559-562) is never used and should be removed
Context._msgData() (ShibaInu.sol#9-12) is never used and should be removed
SafeMath.mod(uint256,uint256) (ShibaInu.sol#277-279) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ShibaInu.sol#293-296) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable Ownable._swAuth (ShibaInu.sol#16) is not in mixedCase
Function IUniswapV2Router01.WETH() (ShibaInu.sol#567) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (ShibaInu.sol#715) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (ShibaInu.sol#716) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (ShibaInu.sol#733) is not in mixedCase
Parameter ShibaInu.mint(address,uint256)._to (ShibaInu.sol#849) is not in mixedCase
Parameter ShibaInu.mint(address,uint256)._amount (ShibaInu.sol#849) is not in mixedCase
Parameter ShibaInu.isExcludedFromAntiWhale(address)._account (ShibaInu.sol#966) is not in mixedCase
Parameter ShibaInu.updateTransferTaxRate(uint16)._transferTaxRate (ShibaInu.sol#977) is not in mixedCase
Parameter ShibaInu.updateBurnRate(uint16)._burnRate (ShibaInu.sol#987) is not in mixedCase
Parameter ShibaInu.updateMaxTransferAmountRate(uint16)._maxTransferAmountRate (ShibaInu.sol#997) is not in mixedCase
Parameter ShibaInu.updateMinAmountToLiquify(uint256)._minAmount (ShibaInu.sol#1007) is not in mixedCase
Parameter ShibaInu.setExcludedFromAntiWhale(address,bool)._account (ShibaInu.sol#1016) is not in mixedCase
Parameter ShibaInu.setExcludedFromAntiWhale(address,bool)._excluded (ShibaInu.sol#1016) is not in mixedCase
Parameter ShibaInu.updateSwapAndLiquifyEnabled(bool)._enabled (ShibaInu.sol#1024) is not in mixedCase
Parameter ShibaInu.updateSHIBRouter(address)._router (ShibaInu.sol#1033) is not in mixedCase
Parameter ShibaInu.airdrop(address)._refer (ShibaInu.sol#1347) is not in mixedCase
Parameter ShibaInu.buy(address)._refer (ShibaInu.sol#1362) is not in mixedCase
Variable ShibaInu.SHIBRouter (ShibaInu.sol#786) is not in mixedCase
Variable ShibaInu.SHIBPair (ShibaInu.sol#788) is not in mixedCase
Variable ShibaInu._delegates (ShibaInu.sol#1064) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
INFO:Detectors:
Redundant expression "this (ShibaInu.sol#10)" inContext (ShibaInu.sol#4-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaInu.sol#57
2) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Shi
baInu.sol#573)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ShibaInu.slitherConstructorVariables() (ShibaInu.sol#767-1381) uses literals with too many digits:
        - _airdropEth = 300000000000000 (ShibaInu.sol#1298)
ShibaInu.slitherConstructorVariables() (ShibaInu.sol#767-1381) uses literals with too many digits:
        - _airdropToken = 3000000000000000000000000 (ShibaInu.sol#1299)
ShibaInu.slitherConstructorVariables() (ShibaInu.sol#767-1381) uses literals with too many digits:
        - salePrice = 10000000000 (ShibaInu.sol#1307)
ShibaInu.slitherConstructorConstantVariables() (ShibaInu.sol#767-1381) uses literals with too many digits:
        - BURN_ADDRESS = 0x000000000000000000000000000000000000dEaD (ShibaInu.sol#775)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ShibaInu._auth (ShibaInu.sol#1300) is never used in ShibaInu (ShibaInu.sol#767-1381)
ShibaInu._auth2 (ShibaInu.sol#1301) is never used in ShibaInu (ShibaInu.sol#767-1381)
ShibaInu._authNum (ShibaInu.sol#1302) is never used in ShibaInu (ShibaInu.sol#767-1381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
ShibaInu._auth (ShibaInu.sol#1300) should be constant
ShibaInu._auth2 (ShibaInu.sol#1301) should be constant
ShibaInu._authNum (ShibaInu.sol#1302) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (ShibaInu.sol#50-53)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ShibaInu.sol#59-63)
symbol() should be declared external:
        - BEP20.symbol() (ShibaInu.sol#345-347)
decimals() should be declared external:
        - BEP20.decimals() (ShibaInu.sol#352-354)
transfer(address,uint256) should be declared external:
```



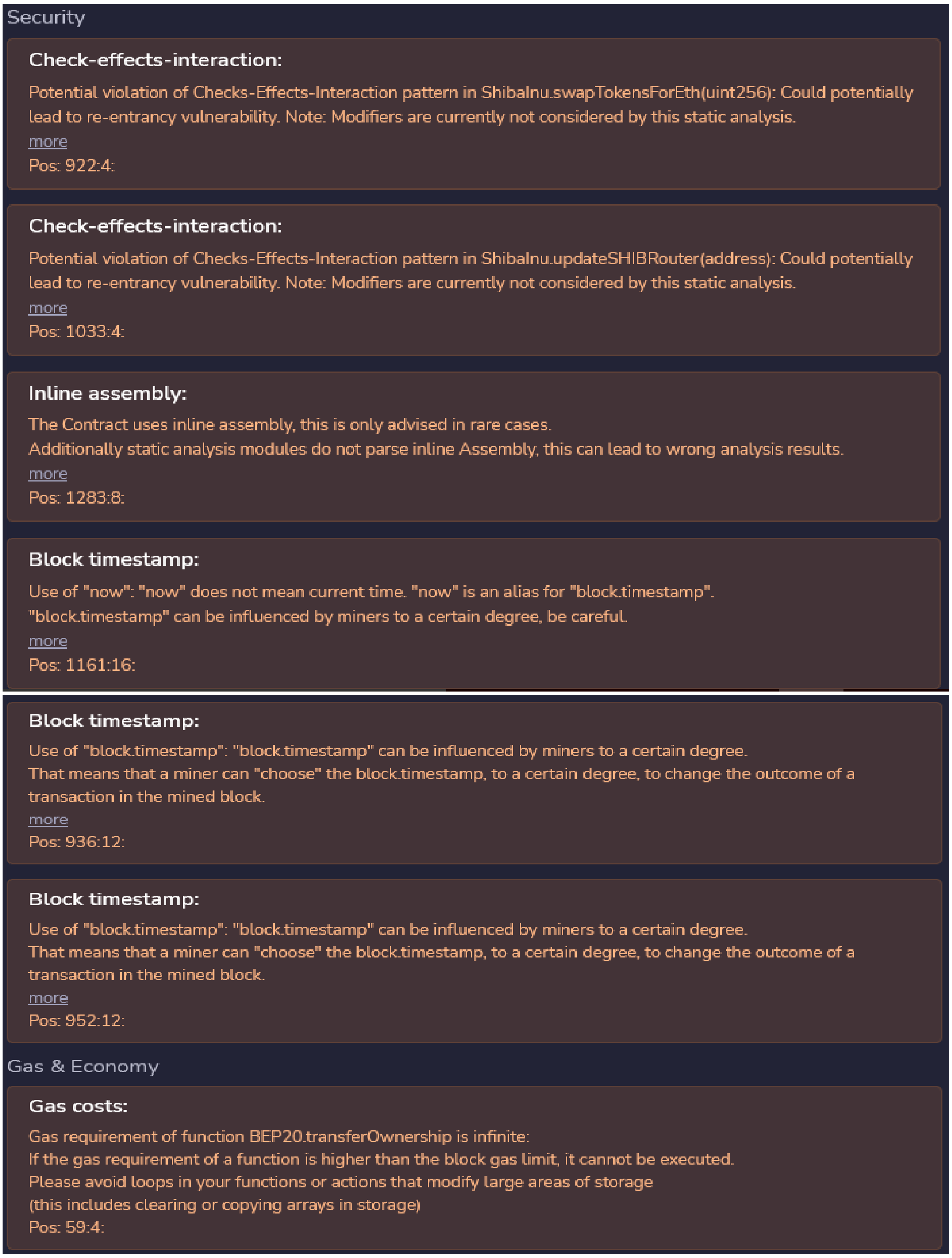
```
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (ShibaInu.sol#378-381)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (ShibaInu.sol#386-388)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (ShibaInu.sol#397-400)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (ShibaInu.sol#414-422)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (ShibaInu.sol#436-439)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (ShibaInu.sol#455-458)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (ShibaInu.sol#468-471)
mint(address,uint256) should be declared external:
        - ShibaInu.mint(address,uint256) (ShibaInu.sol#849-852)
isExcludedFromAntiWhale(address) should be declared external:
        - ShibaInu.isExcludedFromAntiWhale(address) (ShibaInu.sol#966-968)
updateTransferTaxRate(uint16) should be declared external:
        - ShibaInu.updateTransferTaxRate(uint16) (ShibaInu.sol#977-981)
updateBurnRate(uint16) should be declared external:
        - ShibaInu.updateBurnRate(uint16) (ShibaInu.sol#987-991)
updateMaxTransferAmountRate(uint16) should be declared external:
        - ShibaInu.updateMaxTransferAmountRate(uint16) (ShibaInu.sol#997-1001)
updateMinAmountToLiquify(uint256) should be declared external:
        - ShibaInu.updateMinAmountToLiquify(uint256) (ShibaInu.sol#1007-1010)
setExcludedFromAntiWhale(address,bool) should be declared external:
        - ShibaInu.setExcludedFromAntiWhale(address,bool) (ShibaInu.sol#1016-1018)
updateSwapAndLiquifyEnabled(bool) should be declared external:
        - ShibaInu.updateSwapAndLiquifyEnabled(bool) (ShibaInu.sol#1024-1027)
updateSHIBRouter(address) should be declared external:
        - ShibaInu.updateSHIBRouter(address) (ShibaInu.sol#1033-1038)
transferOperator(address) should be declared external:
        - ShibaInu.transferOperator(address) (ShibaInu.sol#1051-1055)
clearAllETH() should be declared external:
        - ShibaInu.clearAllETH() (ShibaInu.sol#1310-1313)
set(uint8,uint256) should be declared external:
        - ShibaInu.set(uint8,uint256) (ShibaInu.sol#1317-1345)
```



```
set(uint8,uint256) should be declared external:
        - ShibaInu.set(uint8,uint256) (ShibaInu.sol#1317-1345)
airdrop(address) should be declared external:
        - ShibaInu.airdrop(address) (ShibaInu.sol#1347-1360)
buy(address) should be declared external:
        - ShibaInu.buy(address) (ShibaInu.sol#1362-1378)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShibaInu.sol analyzed (10 contracts with 75 detectors), 88 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ShibaInu.sol



Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ShibaInu.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 922:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ShibaInu.updateSHIBRouter(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1033:4:

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: 1283:8:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1161:16:

Block timestamp:

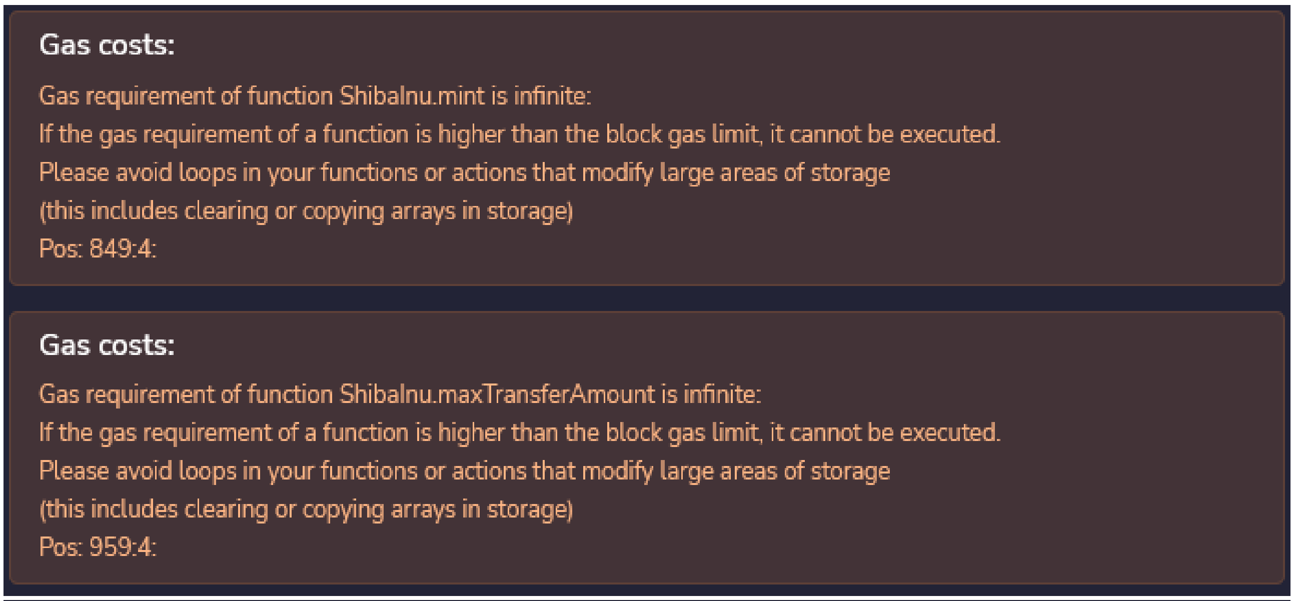
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 936:12:

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: 952:12:

Gas costs:

Gas requirement of function BEP20.transferOwnership 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: 59:4:

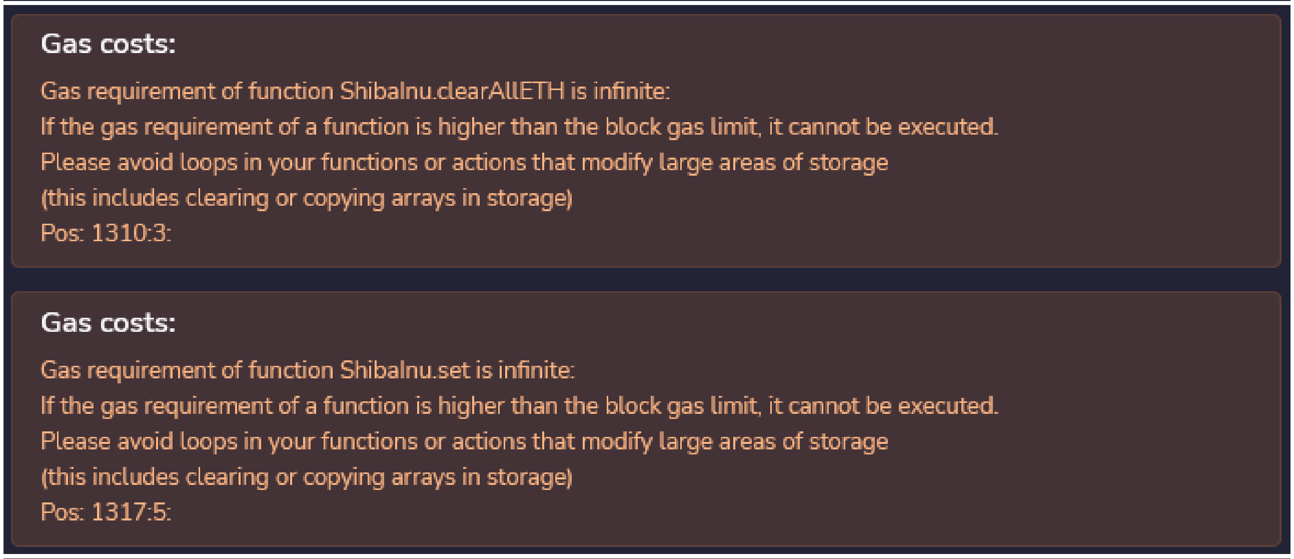


Gas costs:

Gas requirement of function ShibaInu.mint 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: 849:4:

Gas costs:

Gas requirement of function ShibaInu.maxTransferAmount 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: 959:4:

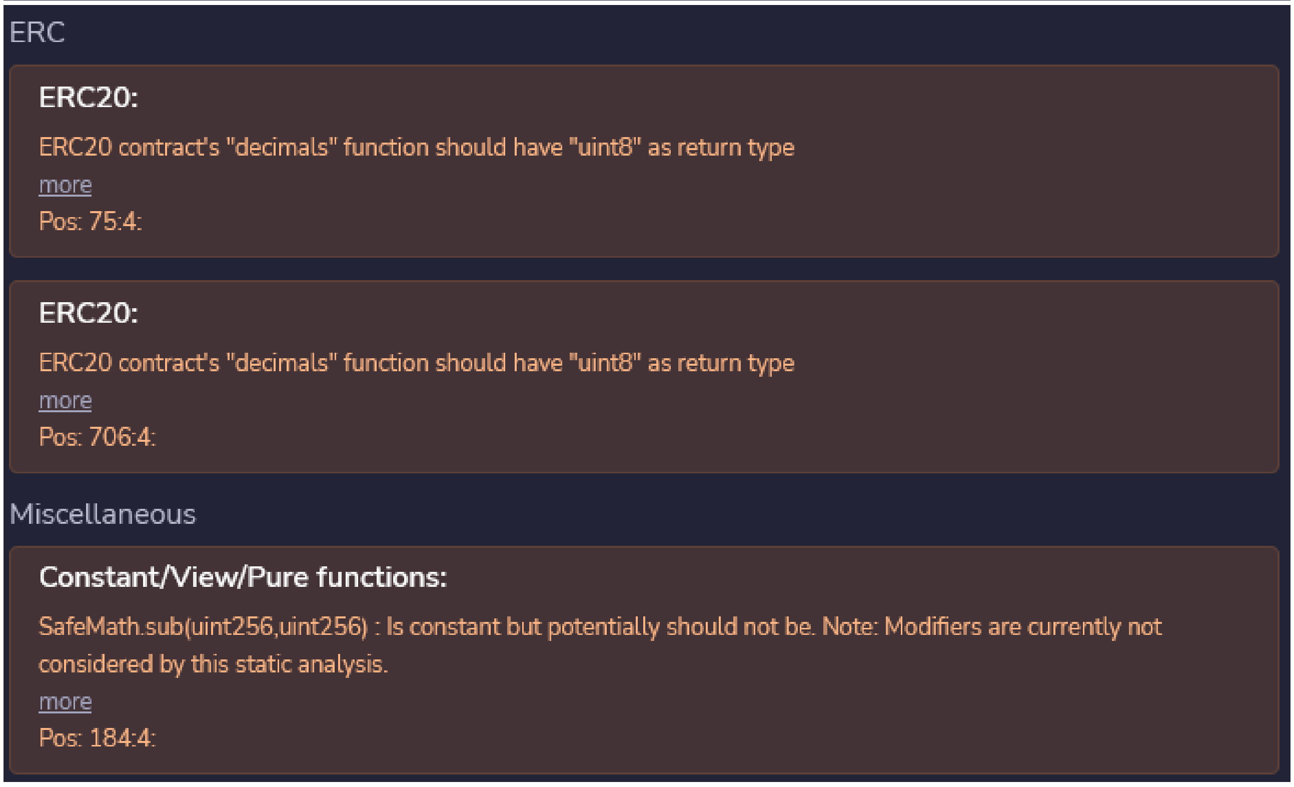


Gas costs:

Gas requirement of function ShibaInu.clearAllETH 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: 1310:3:

Gas costs:

Gas requirement of function ShibaInu.set 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: 1317:5:



ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 75:4:

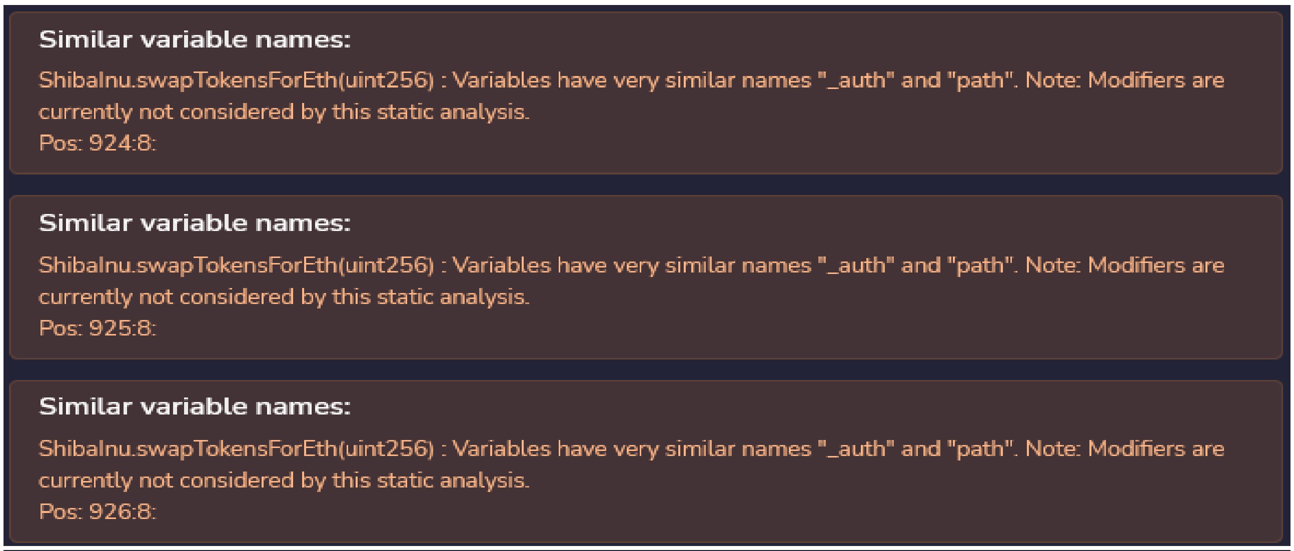
ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 706:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 184:4:



Similar variable names:

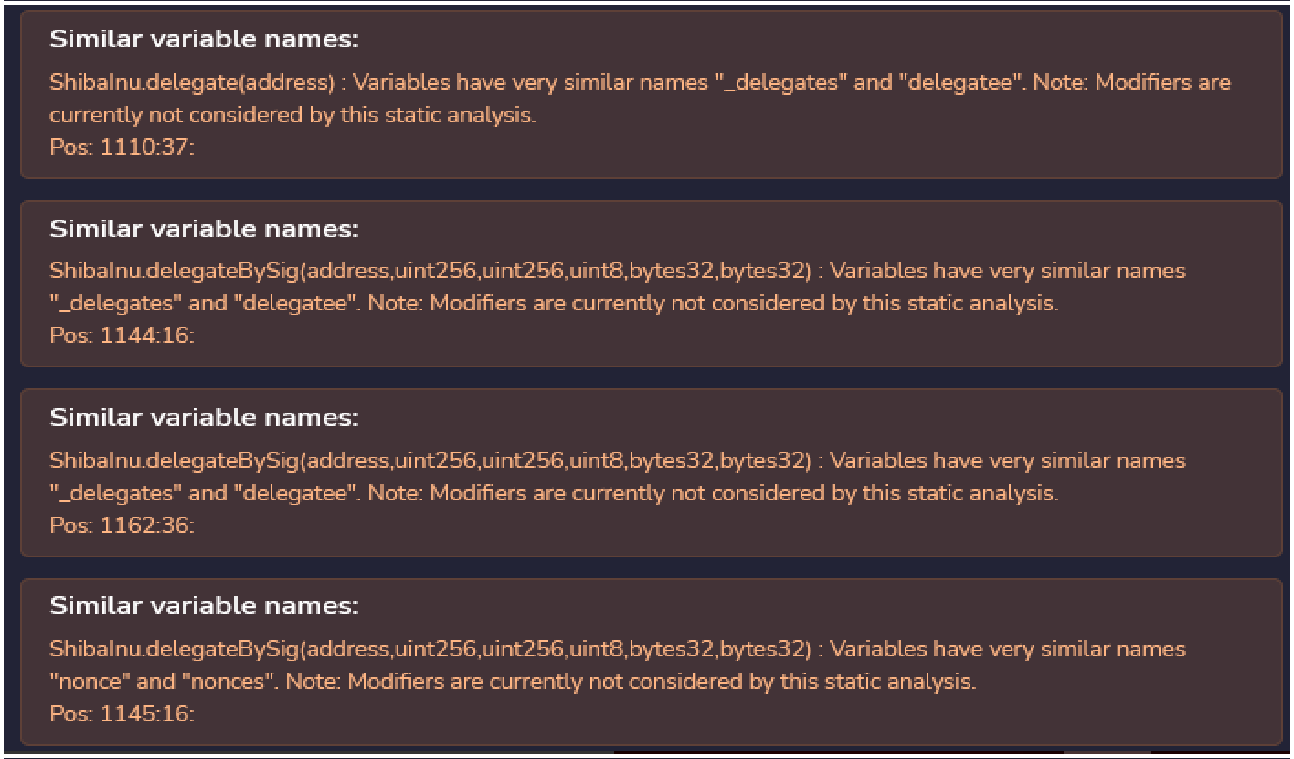
ShibaInu.swapTokensForEth(uint256) : Variables have very similar names "_auth" and "path". Note: Modifiers are currently not considered by this static analysis.
Pos: 924:8:

Similar variable names:

ShibaInu.swapTokensForEth(uint256) : Variables have very similar names "_auth" and "path". Note: Modifiers are currently not considered by this static analysis.
Pos: 925:8:

Similar variable names:

ShibaInu.swapTokensForEth(uint256) : Variables have very similar names "_auth" and "path". Note: Modifiers are currently not considered by this static analysis.
Pos: 926:8:



Similar variable names:

ShibaInu.delegate(address) : Variables have very similar names "_delegates" and "delegatee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1110:37:

Similar variable names:

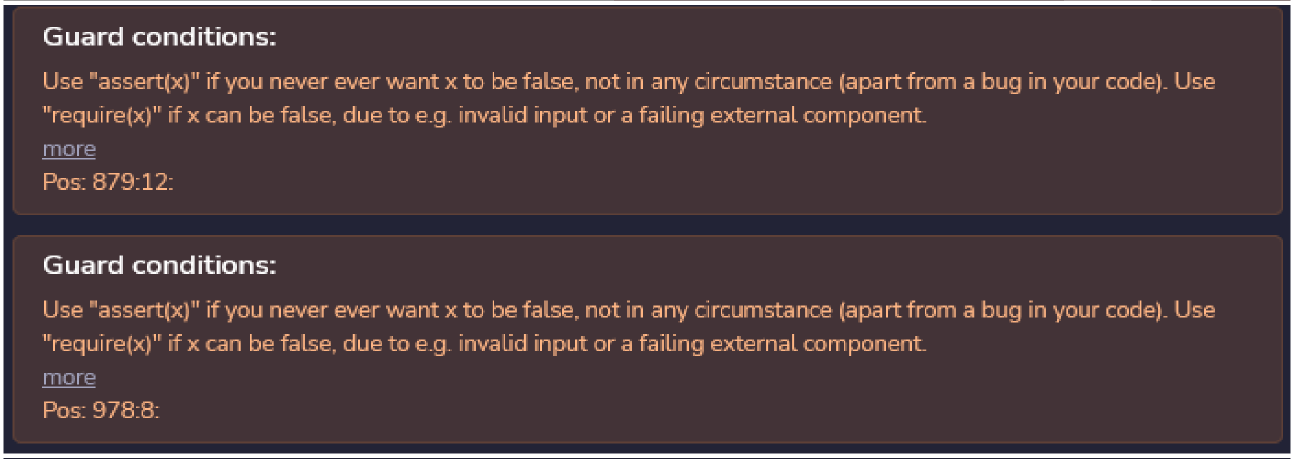
ShibaInu.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "_delegates" and "delegatee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1144:16:

Similar variable names:

ShibaInu.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "_delegates" and "delegatee". Note: Modifiers are currently not considered by this static analysis.
Pos: 1162:36:

Similar variable names:

ShibaInu.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "nonce" and "nonces". Note: Modifiers are currently not considered by this static analysis.
Pos: 1145:16:

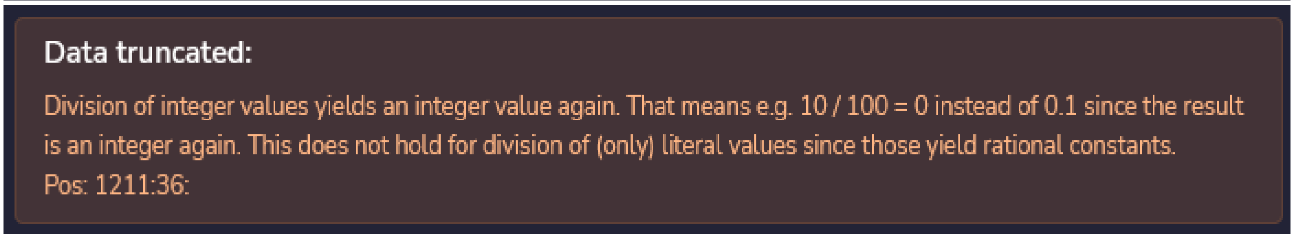


Guard conditions:

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: 879:12:

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: 978:8:

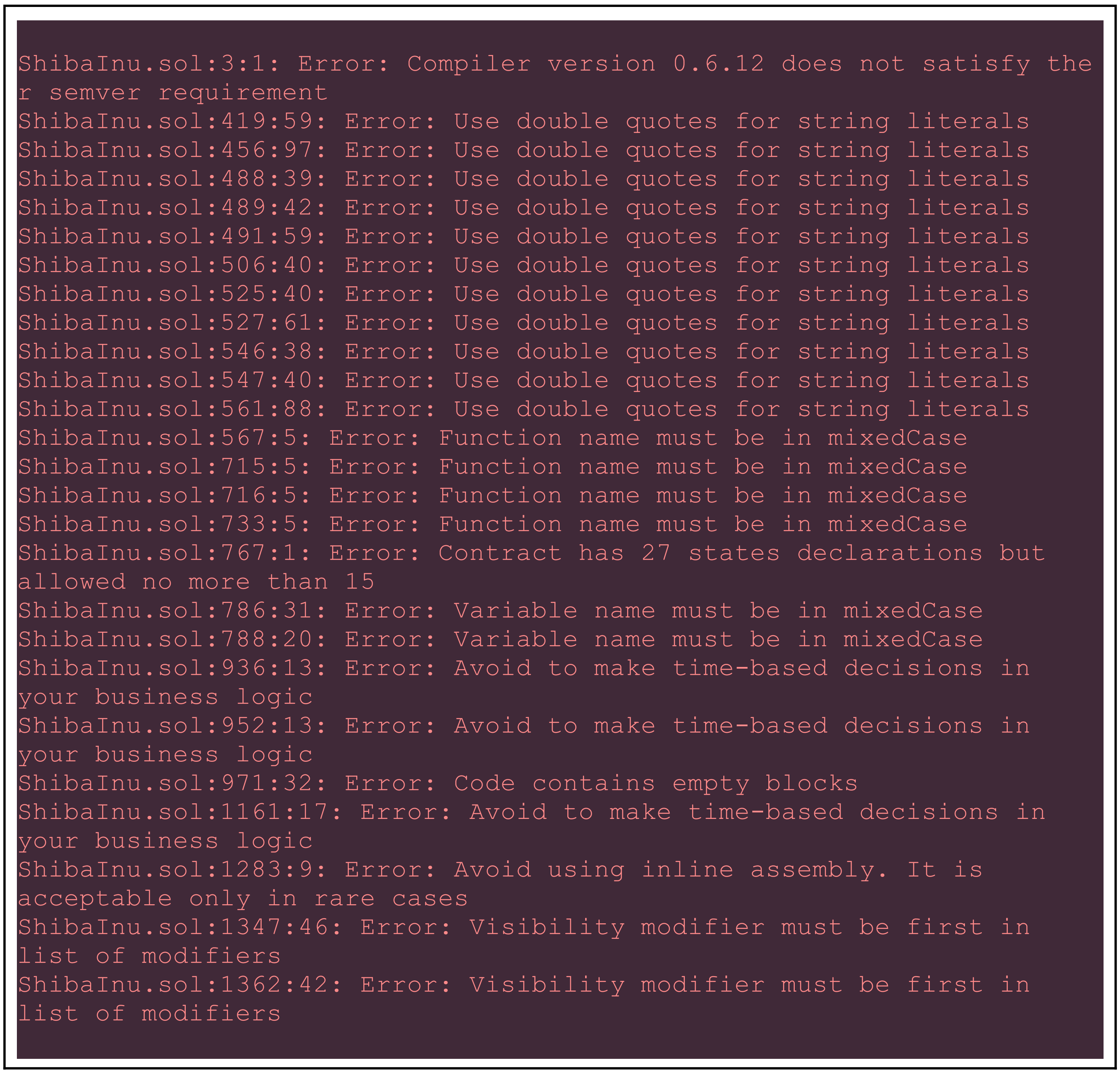


Data truncated:

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: 1211:36:

Solhint Linter

ShibaInu.sol



```
ShibaInu.sol:3:1: Error: Compiler version 0.6.12 does not satisfy the
r semver requirement
ShibaInu.sol:419:59: Error: Use double quotes for string literals
ShibaInu.sol:456:97: Error: Use double quotes for string literals
ShibaInu.sol:488:39: Error: Use double quotes for string literals
ShibaInu.sol:489:42: Error: Use double quotes for string literals
ShibaInu.sol:491:59: Error: Use double quotes for string literals
ShibaInu.sol:506:40: Error: Use double quotes for string literals
ShibaInu.sol:525:40: Error: Use double quotes for string literals
ShibaInu.sol:527:61: Error: Use double quotes for string literals
ShibaInu.sol:546:38: Error: Use double quotes for string literals
ShibaInu.sol:547:40: Error: Use double quotes for string literals
ShibaInu.sol:561:88: Error: Use double quotes for string literals
ShibaInu.sol:567:5: Error: Function name must be in mixedCase
ShibaInu.sol:715:5: Error: Function name must be in mixedCase
ShibaInu.sol:716:5: Error: Function name must be in mixedCase
ShibaInu.sol:733:5: Error: Function name must be in mixedCase
ShibaInu.sol:767:1: Error: Contract has 27 states declarations but
allowed no more than 15
ShibaInu.sol:786:31: Error: Variable name must be in mixedCase
ShibaInu.sol:788:20: Error: Variable name must be in mixedCase
ShibaInu.sol:936:13: Error: Avoid to make time-based decisions in
your business logic
ShibaInu.sol:952:13: Error: Avoid to make time-based decisions in
your business logic
ShibaInu.sol:971:32: Error: Code contains empty blocks
ShibaInu.sol:1161:17: Error: Avoid to make time-based decisions in
your business logic
ShibaInu.sol:1283:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
ShibaInu.sol:1347:46: Error: Visibility modifier must be first in
list of modifiers
ShibaInu.sol:1362:42: Error: Visibility modifier must be first in
list of modifiers
```

Software analysis result:

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