Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	ShibaMask
Platform:	Binance Smart Chain
Website:	shibamask.org
Language:	Solidity
Date:	October 28th, 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 ShibaMask team to perform the Security audit of the ShibaMask 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 28th, 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

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

Audit scope

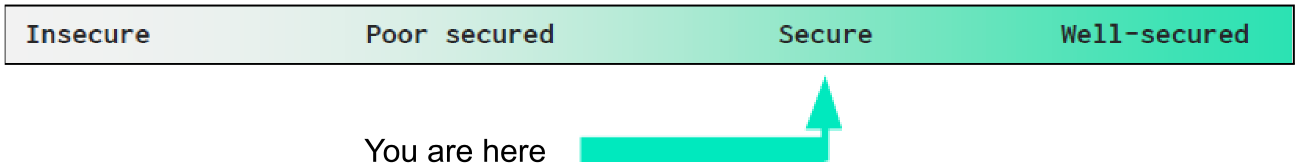
Name	**Code Review and Security Analysis Report for ShibaMask Token Smart Contract**
Platform	**BSC / Solidity**
File	shibapy.sol
File MD5 Hash	3433551CF329C1D7C5A23D845D2B63FD
Online code	0x1c384884637099fc002730389aa3760a3c886fee
Audit Date	October 28th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: ShibaMask ● Symbol: SHBMA ● Decimals: 4	**YES, This is valid.**
● Buy Tax Fee: 0 ● Buy Liquidity Fee: 0 ● Buy Burn Fee: 0 ● Buy Marketing Fee: 0 ● Sell Tax Fee: 1% ● Sell Liquidity Fee: 8% ● Sell Burn Fee: 1% ● Sell Marketing Fee: 1% ● Transfer Tax Fee : 0 ● Transfer Liquidity Fee: 0 ● Transfer Burn Fee: 0 ● Transfer Marketing Fee: 0 ● Maximum Transaction Amount: 1 Trillion SHBMA ● Number Tokens Sell To Add To Liquidity: 50 Billion SHBMA	**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	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	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 ShibaMask 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 ShibaMask Token.

The ShibaMask 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 ShibaMask 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.

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	read	Passed	No Issue
15	totalFees	read	Passed	No Issue
16	totalBurn	read	Passed	No Issue
17	addInblacklist	write	access only Owner	No Issue
18	removeFromblacklist	write	access only Owner	No Issue
19	checkBlacklist	read	Passed	No Issue
20	deliver	write	Function input parameters lack of check	Refer Audit Findings
21	reflectionFromToken	read	Passed	No Issue
22	tokenFromReflection	read	Passed	No Issue
23	excludeFromReward	write	access only Owner	No Issue
24	includeInReward	external	Infinite loops, Out of GAs issue	Refer Audit Findings
25	burn	write	access only Owner	No Issue
26	_burn	internal	Passed	No Issue
27	excludeFromFee	write	access only Owner	No Issue
28	includeInFee	write	access only Owner	No Issue
29	changeMarketingWallet	write	Function input parameters lack of check	Refer Audit Findings

30	setMinLiquidityPercent	external	Function input parameters lack of check	Refer Audit Findings
31	setMaxTxPercent	external	Function input parameters lack of check	Refer Audit Findings
32	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
33	receive	external	Passed	No Issue
34	rescueBNBFromContract	external	access only Owner	No Issue
35	transferAnyBEP20Tokens	write	access only Owner	No Issue
36	updateNumTokensSellToAddToLiquidity	external	Function input parameters lack of check	Refer Audit Findings
37	setUniswapRouter	external	Function input parameters lack of check	Refer Audit Findings
38	setUniswapPair	external	Function input parameters lack of check	Refer Audit Findings
39	setExcludedFromAutoLiquidity	external	Function input parameters lack of check	Refer Audit Findings
40	setExcludedToAutoLiquidity	external	Function input parameters lack of check	Refer Audit Findings
41	removeAllFee	write	Passed	No Issue
42	restoreAllFee	write	Passed	No Issue
43	isExcludedFromFee	read	Passed	No Issue
44	_approve	write	Passed	No Issue
45	_transfer	write	Passed	No Issue
46	swapAndLiquify	write	access by lockTheSwap	No Issue
47	swapTokensForEth	write	Passed	No Issue
48	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
49	_tokenTransfer	write	Passed	No Issue
50	_transferBothExcluded	write	Passed	No Issue
51	_transferStandard	write	Passed	No Issue
52	_transferToExcluded	write	Passed	No Issue
53	_transferFromExcluded	write	Passed	No Issue
54	_reflectFee	write	Passed	No Issue
55	_getTValues	read	Passed	No Issue
56	_getRValues	write	Passed	No Issue
57	_getRate	write	Passed	No Issue
58	_getCurrentSupply	read	Infinite loops, Out of GAs issue	Refer Audit Findings

59	_takeLiquidity	write	Passed	No Issue
60	calculateTaxFee	read	Passed	No Issue
61	calculateLiquidityFee	read	Passed	No Issue
62	calculateBurnFee	read	Passed	No Issue
63	calculateMarketingFee	read	Passed	No Issue
64	sendtoMarktingWallet	write	Passed	No Issue
65	changeBuyFee	write	Function input parameters lack of check	Refer Audit Findings
66	changeSellFee	write	Function input parameters lack of check	Refer Audit Findings
67	changeTransferFee	write	Function input parameters lack of check	Refer Audit Findings
68	_pause	write	access only Owner	No Issue
69	unpause	write	Missing required error message	Refer Audit Findings
70	_msgSender	internal	Passed	No Issue
71	_msgData	internal	Passed	No Issue
72	owner	read	Passed	No Issue
73	onlyOwner	modifier	Passed	No Issue
74	renounceOwnership	write	Possible to gain ownership	Refer Audit Findings
75	transferOwnership	write	access only Owner	No Issue
76	geUnlockTime	read	Passed	No Issue
77	lock	write	Possible to gain ownership	Refer Audit Findings
78	unlock	write	Possible to gain ownership	Refer Audit Findings

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) 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.

(2) Function input parameters lack of check:

Some functions require validation before execution.

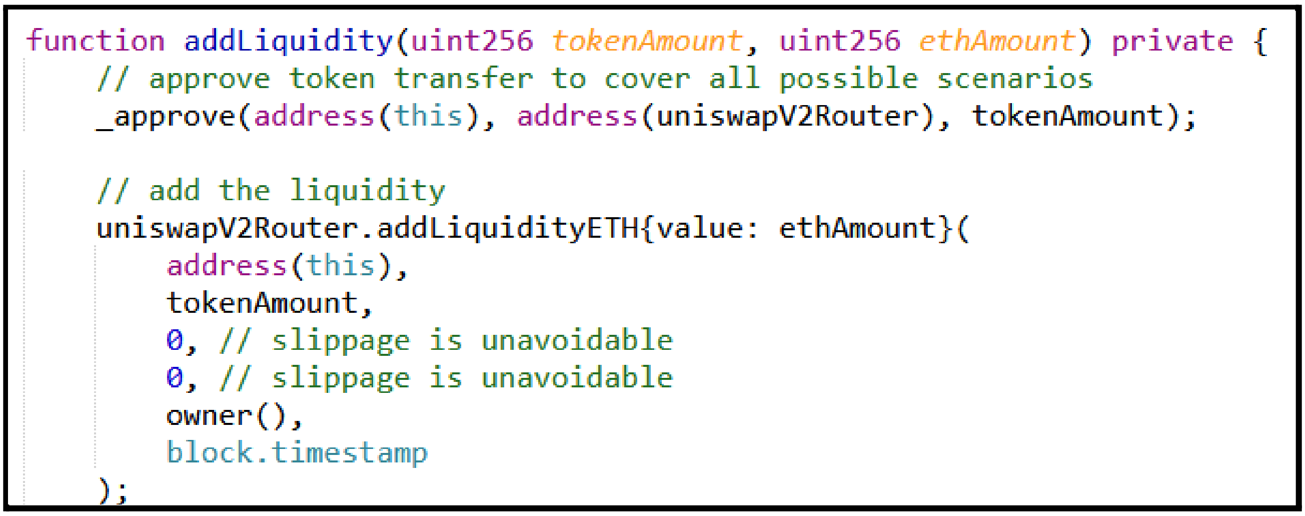
Functions are:

- deliver() = tAmount
- changeMarketingWallet() = wallet
- setMinLiquidityPercent() = minLiquidityPercent
- setMaxTxPercent() = maxTxPercent
- updateNumTokensSellToAddToLiquidity() = amount
- setUniswapRouter() = r
- setUniswapPair() = p
- setExcludedFromAutoLiquidity() = a

- setExcludedToAutoLiquidity() = a
- changeBuyFee() = (buyTax, buyLiquidity, buyBurn, buyMarketing)
- changeSellFee() = (sellTax, sellLiquidity, sellBurn, sellMarketing)
- changeTransferFee() = (transferTax, transferLiquidity, transferBurn, transferMarketing)

Resolution: Use validation : variable should be greater than 0 and for address type check variable is not address(0).

(3) Centralized risk in addLiquidity:



```
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
    );
```

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

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

(4) Possible to gain ownership:

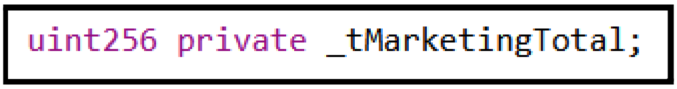
Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

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

Resolution: We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership before calling the lock function.

Very Low / Informational / Best practices:

(1) Unused variables:

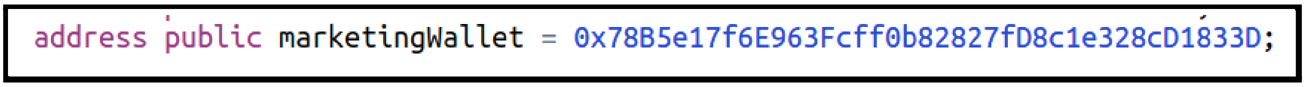


```
uint256 private _tMarketingTotal;
```

There is _tMarketingTotal variable defined but not used anywhere.

Resolution: Remove unused variables from the code.

(2) Hard coded Values:

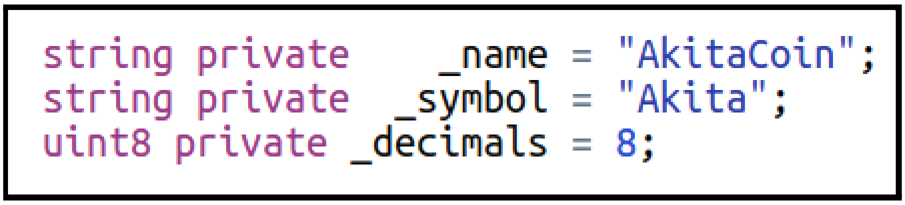


```
address public marketingWallet = 0x78B5e17f6E963Fcff0b82827fD8c1e328cD1833D;
```

Some variables are set as hard coded addresses.

Resolution: Deployer has to confirm before deploying the contract to production.

(3) Make variables constant:

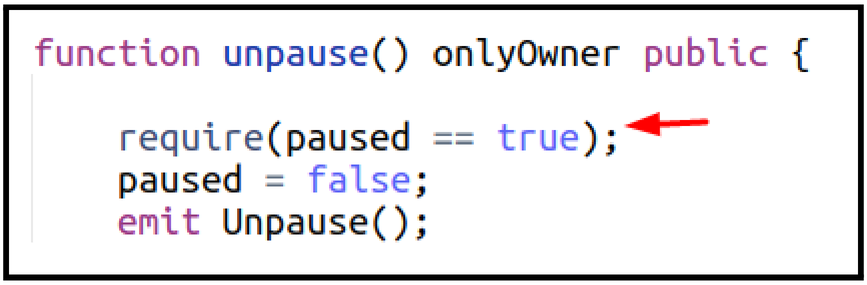


```
string private     _name = "AkitaCoin";
string private   _symbol = "Akita";
uint8 private _decimals = 8;
```

These variables will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare those variables as constant. Just put a constant keyword.

(4) Missing required error message:



```
function unpause() onlyOwner public {

    require(paused == true);
    paused = false;
    emit Unpause();
```

There is no error message require.

Resolution: We suggest setting relevant error message to identify the failure of the transaction.

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:

- addInblacklist: Owner can add address account in block list.
- removeFromblacklist: Owner can remove account from black list.
- excludeFromReward: Owner can check if the account is already excluded or not.
- includeInReward: Owner can include in reward.
- burn: Owner can burn the amount.
- excludeFromFee: Owner can exclude from fee.
- includeInFee: Owner can include in fee.
- changeMarketingWallet: Owner can change marketing wallet.
- setMinLiquidityPercent: Owner can set minimum liquidity percentage.
- setMaxTxPercent: Owner can set maximum percentage.
- setSwapAndLiquifyEnabled: Owner can set swap and liquify enabled status.
- rescueBNBFromContract: Owner can rescue BNB from contract.
- transferAnyBEP20Tokens: Owner cannot transfer out catecoin from this smart contract.
- updateNumTokensSellToAddToLiquidity: Owner can update Number token sell and add to liquidity.
- setUniswapRouter: Owner can set uniswap router address.
- setUniswapPair: Owner can set uniswap pair address.
- setExcludedFromAutoLiquidity: Owner can set the excluded from auto liquidity.
- setExcludedToAutoLiquidity: Owner can set the excluded to auto liquidity.
- changeBuyFee: Owner can change buy fee.
- changeSellFee: Owner can change sell fee.
- changeTransferFee: Owner can change transfer fee.
- _pause: Owner can pause true status.
- unpause: Owner can pause false status.

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).

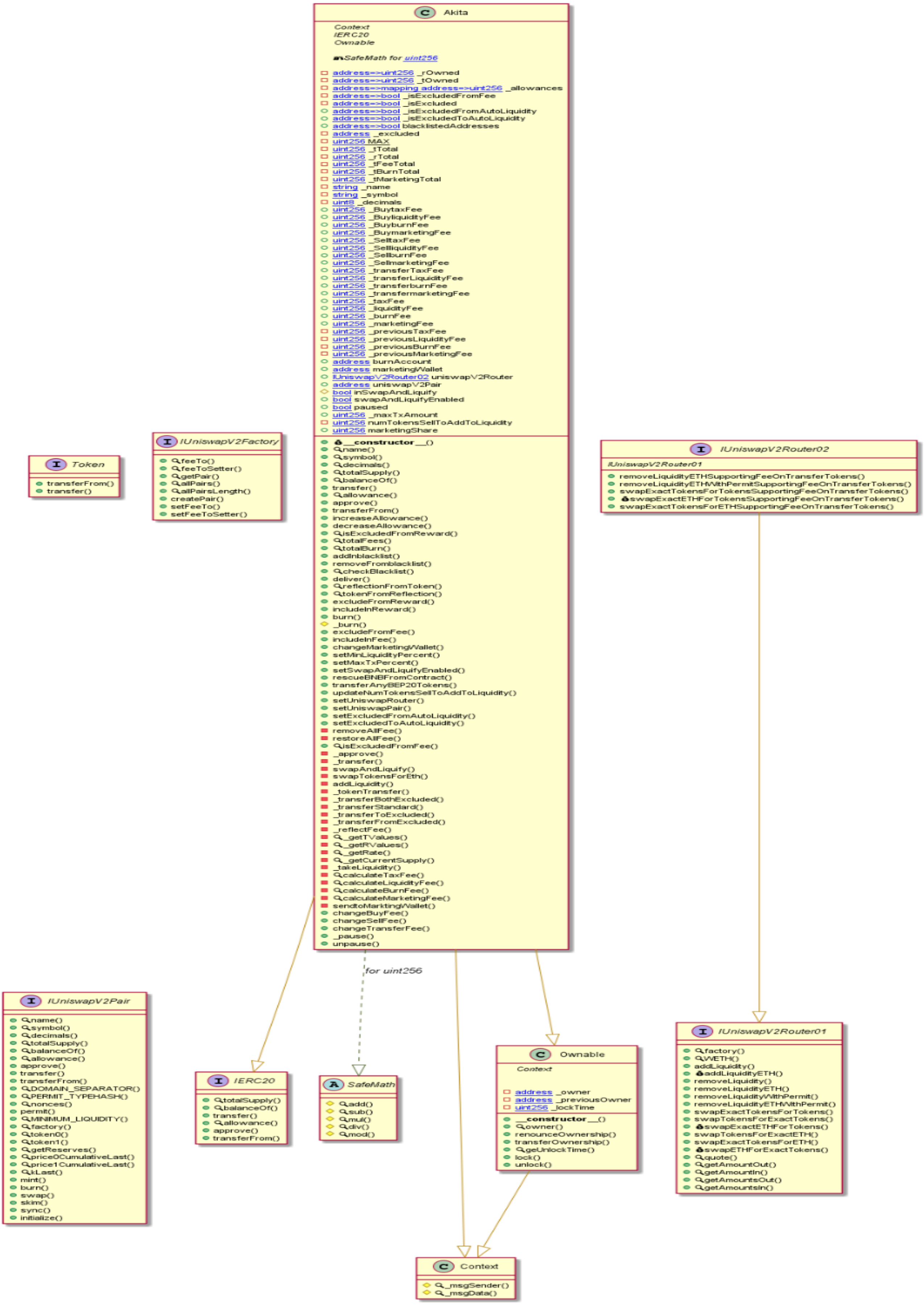
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

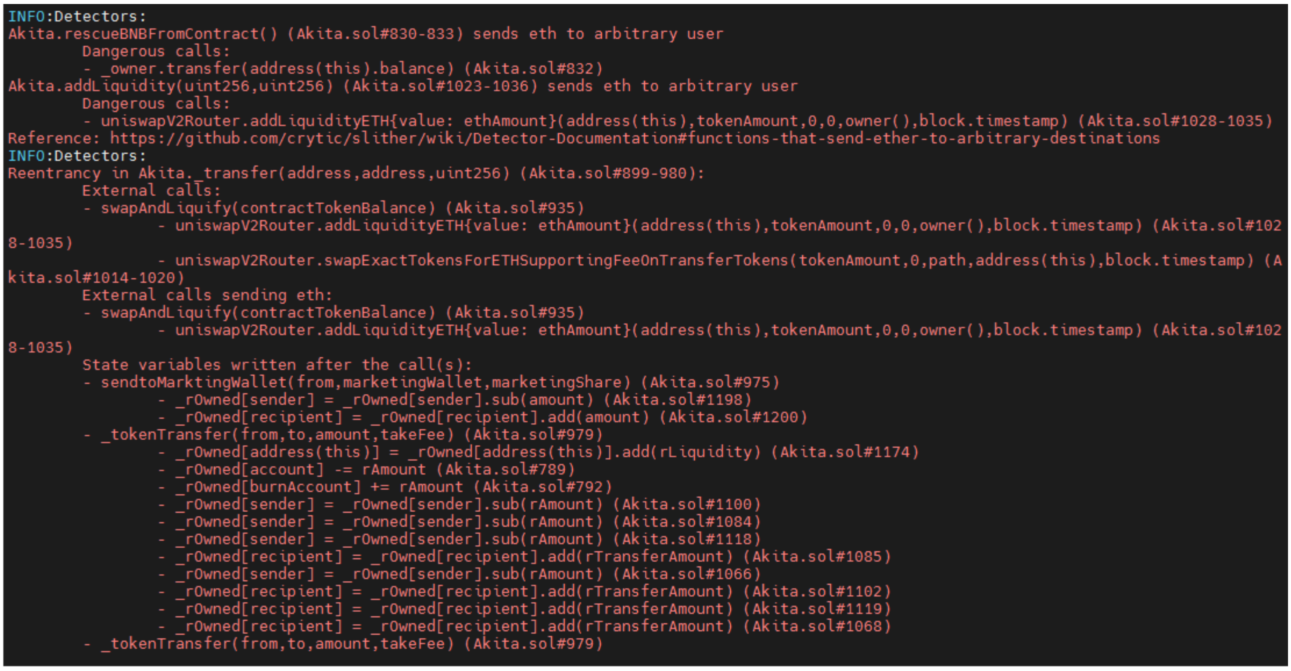
Appendix

Code Flow Diagram - ShibaMask Token

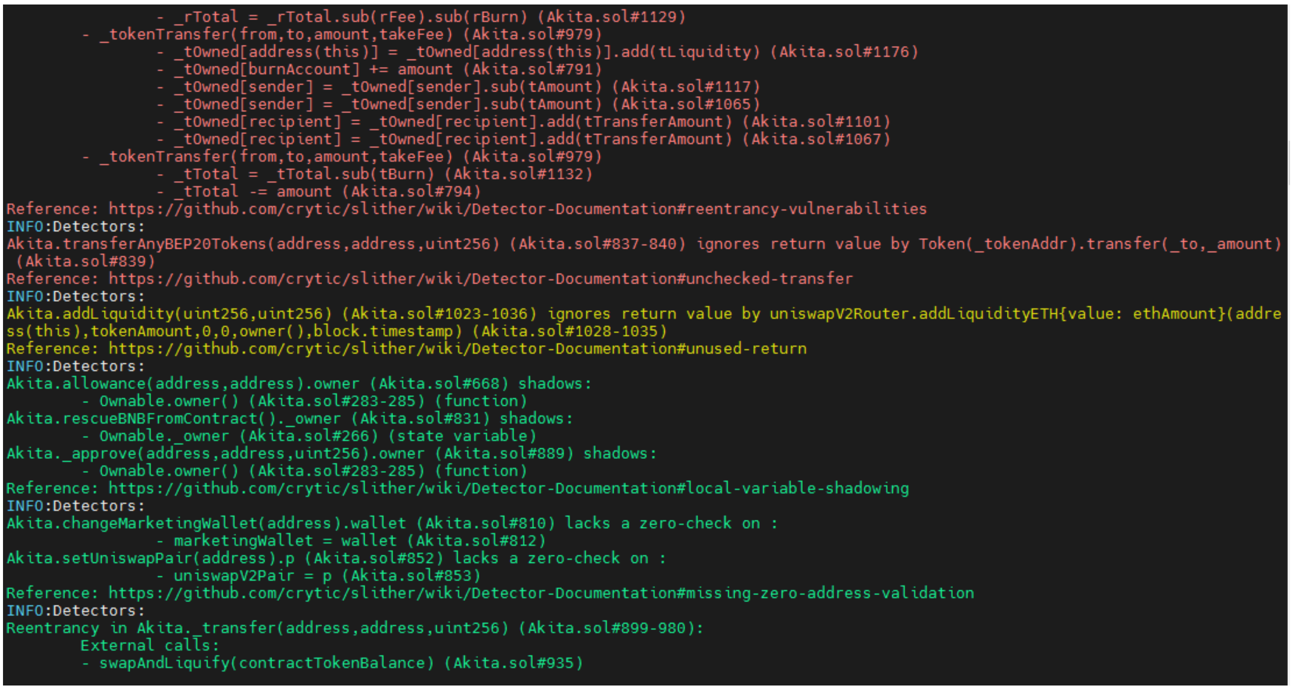


Slither Results Log

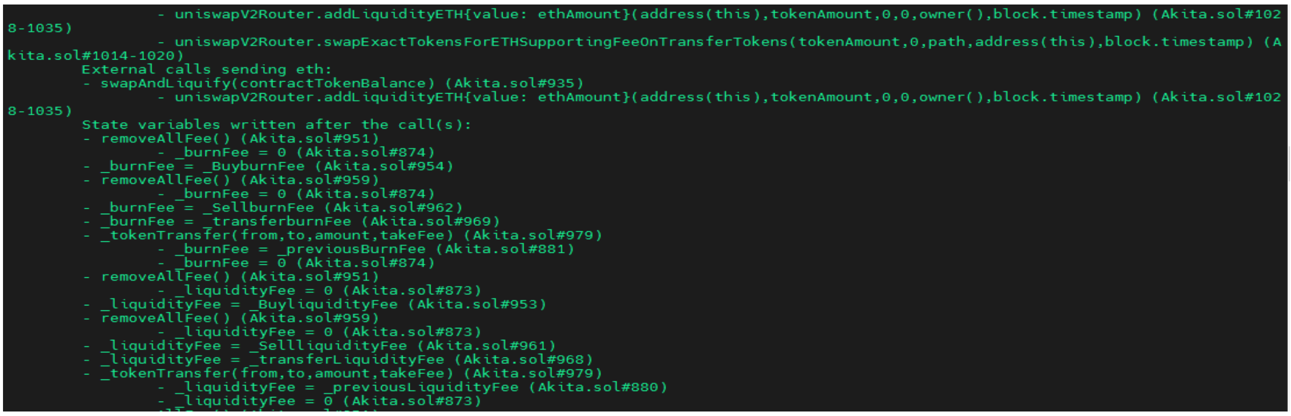
Slither log >> shibapy.sol



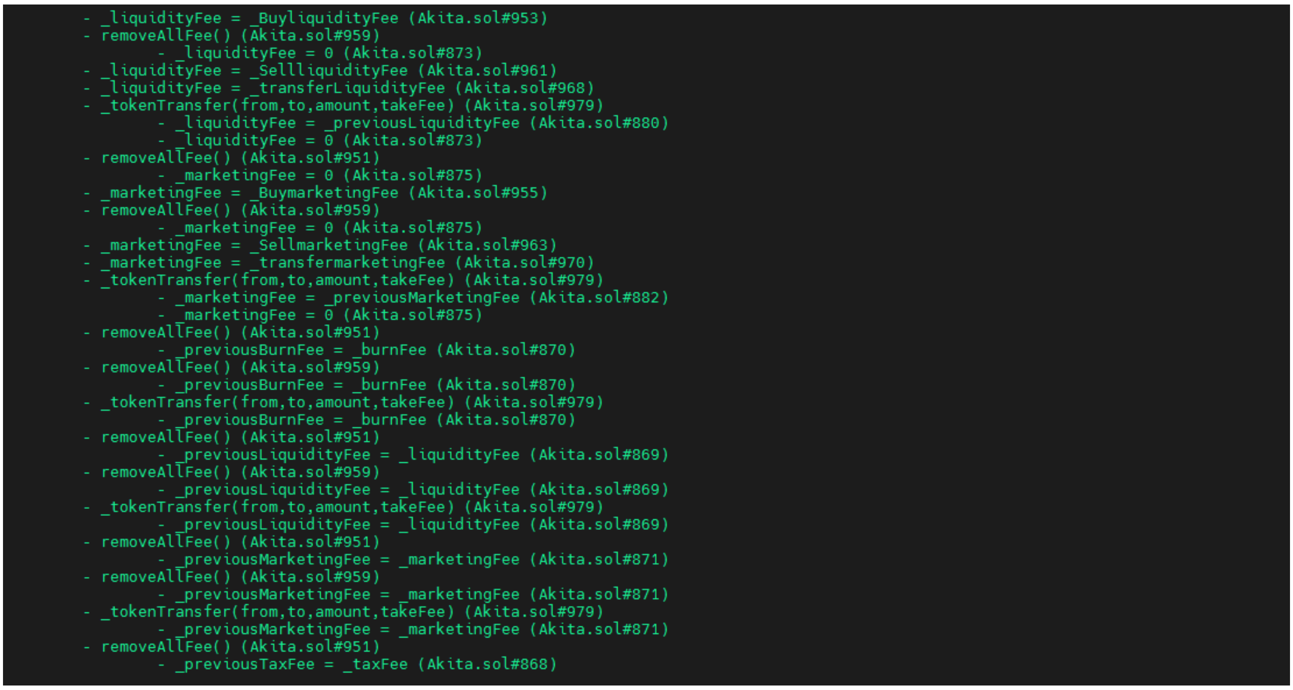
```
INFO:Detectors:
Akita.rescueBNBFromContract() (Akita.sol#830-833) sends eth to arbitrary user
        Dangerous calls:
        - _owner.transfer(address(this).balance) (Akita.sol#832)
Akita.addLiquidity(uint256,uint256) (Akita.sol#1023-1036) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#1028-1035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Akita._transfer(address,address,uint256) (Akita.sol#899-980):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        State variables written after the call(s):
        - sendtoMarktingWallet(from,marketingWallet,marketingShare) (Akita.sol#975)
                - _rOwned[sender] = _rOwned[sender].sub(amount) (Akita.sol#1198)
                - _rOwned[recipient] = _rOwned[recipient].add(amount) (Akita.sol#1200)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (Akita.sol#1174)
                - _rOwned[account] -= rAmount (Akita.sol#789)
                - _rOwned[burnAccount] += rAmount (Akita.sol#792)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Akita.sol#1100)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Akita.sol#1084)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Akita.sol#1118)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Akita.sol#1085)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Akita.sol#1066)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Akita.sol#1102)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Akita.sol#1119)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Akita.sol#1068)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
```



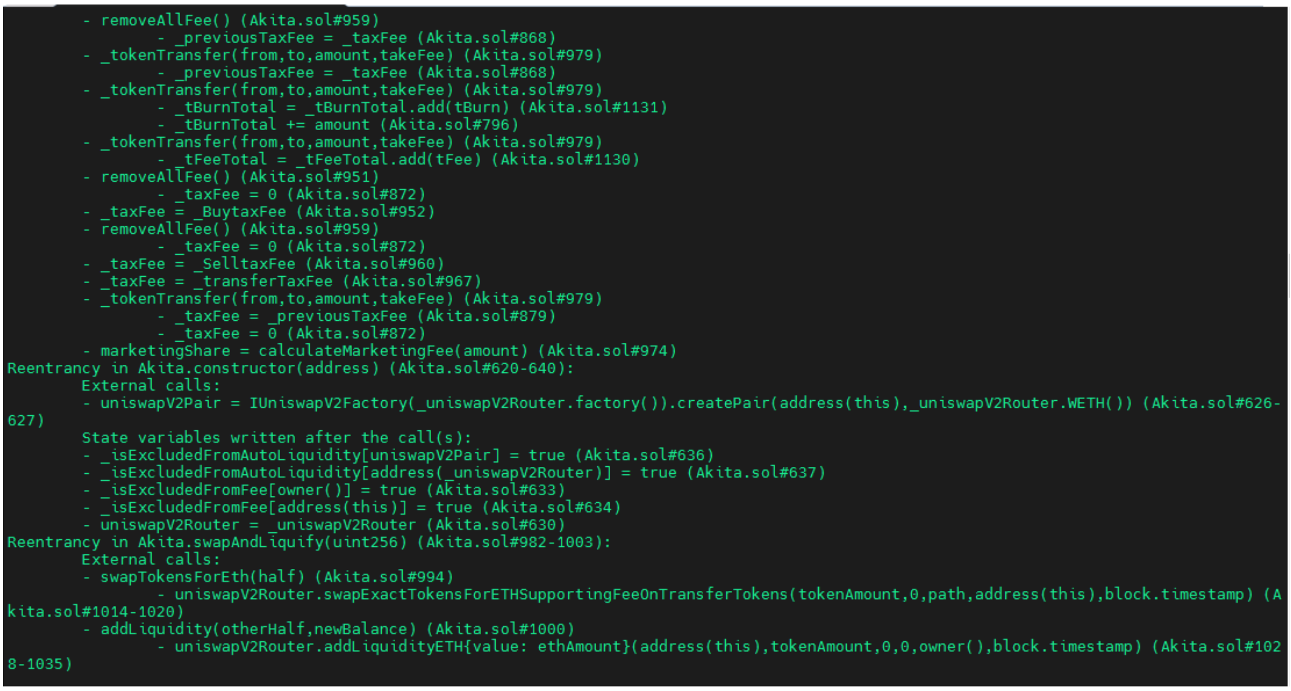
```
                - _rTotal = _rTotal.sub(rFee).sub(rBurn) (Akita.sol#1129)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (Akita.sol#1176)
                - _tOwned[burnAccount] += amount (Akita.sol#791)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Akita.sol#1117)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Akita.sol#1065)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Akita.sol#1101)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Akita.sol#1067)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _tTotal = _tTotal.sub(tBurn) (Akita.sol#1132)
                - _tTotal -= amount (Akita.sol#794)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
Akita.transferAnyBEP20Tokens(address,address,uint256) (Akita.sol#837-840) ignores return value by Token(_tokenAddr).transfer(_to,_amount)
 (Akita.sol#839)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Akita.addLiquidity(uint256,uint256) (Akita.sol#1023-1036) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(addre
ss(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#1028-1035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Akita.allowance(address,address).owner (Akita.sol#668) shadows:
        - Ownable.owner() (Akita.sol#283-285) (function)
Akita.rescueBNBFromContract()._owner (Akita.sol#831) shadows:
        - Ownable._owner (Akita.sol#266) (state variable)
Akita._approve(address,address,uint256).owner (Akita.sol#889) shadows:
        - Ownable.owner() (Akita.sol#283-285) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Akita.changeMarketingWallet(address).wallet (Akita.sol#810) lacks a zero-check on :
                - marketingWallet = wallet (Akita.sol#812)
Akita.setUniswapPair(address).p (Akita.sol#852) lacks a zero-check on :
                - uniswapV2Pair = p (Akita.sol#853)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Akita._transfer(address,address,uint256) (Akita.sol#899-980):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
```



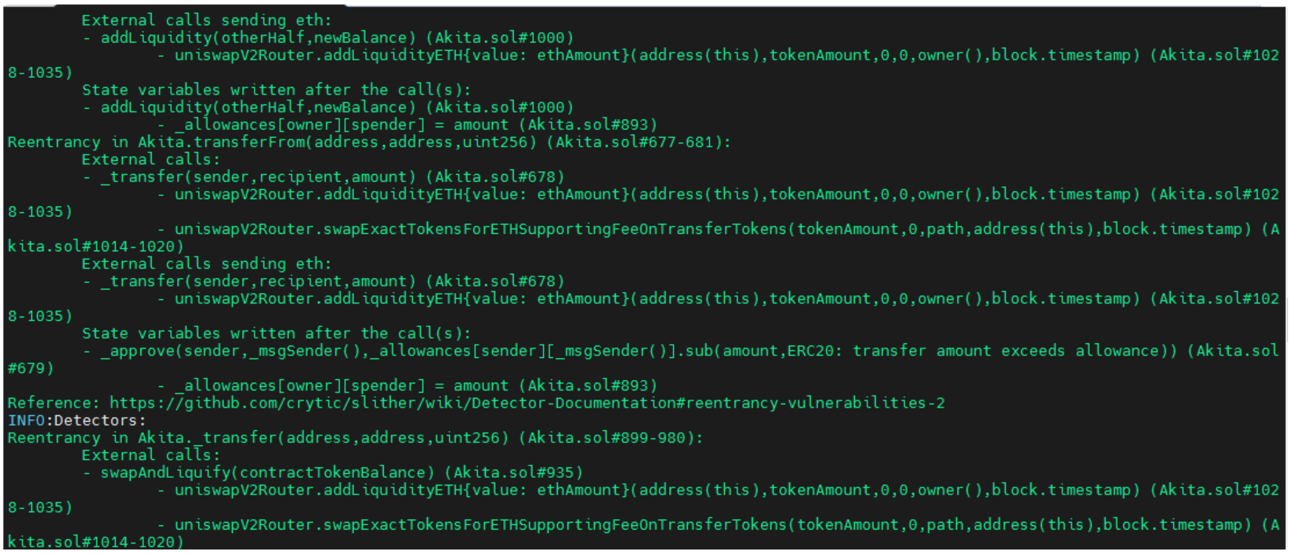
```
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        State variables written after the call(s):
        - removeAllFee() (Akita.sol#951)
                - _burnFee = 0 (Akita.sol#874)
        - _burnFee = _BuyburnFee (Akita.sol#954)
        - removeAllFee() (Akita.sol#959)
                - _burnFee = 0 (Akita.sol#874)
        - _burnFee = _SellburnFee (Akita.sol#962)
        - _burnFee = _transferburnFee (Akita.sol#969)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _burnFee = _previousBurnFee (Akita.sol#881)
                - _burnFee = 0 (Akita.sol#874)
        - removeAllFee() (Akita.sol#951)
                - _liquidityFee = 0 (Akita.sol#873)
        - _liquidityFee = _BuyliquidityFee (Akita.sol#953)
        - removeAllFee() (Akita.sol#959)
                - _liquidityFee = 0 (Akita.sol#873)
        - _liquidityFee = _SellliquidityFee (Akita.sol#961)
        - _liquidityFee = _transferLiquidityFee (Akita.sol#968)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _liquidityFee = _previousLiquidityFee (Akita.sol#880)
                - _liquidityFee = 0 (Akita.sol#873)
```



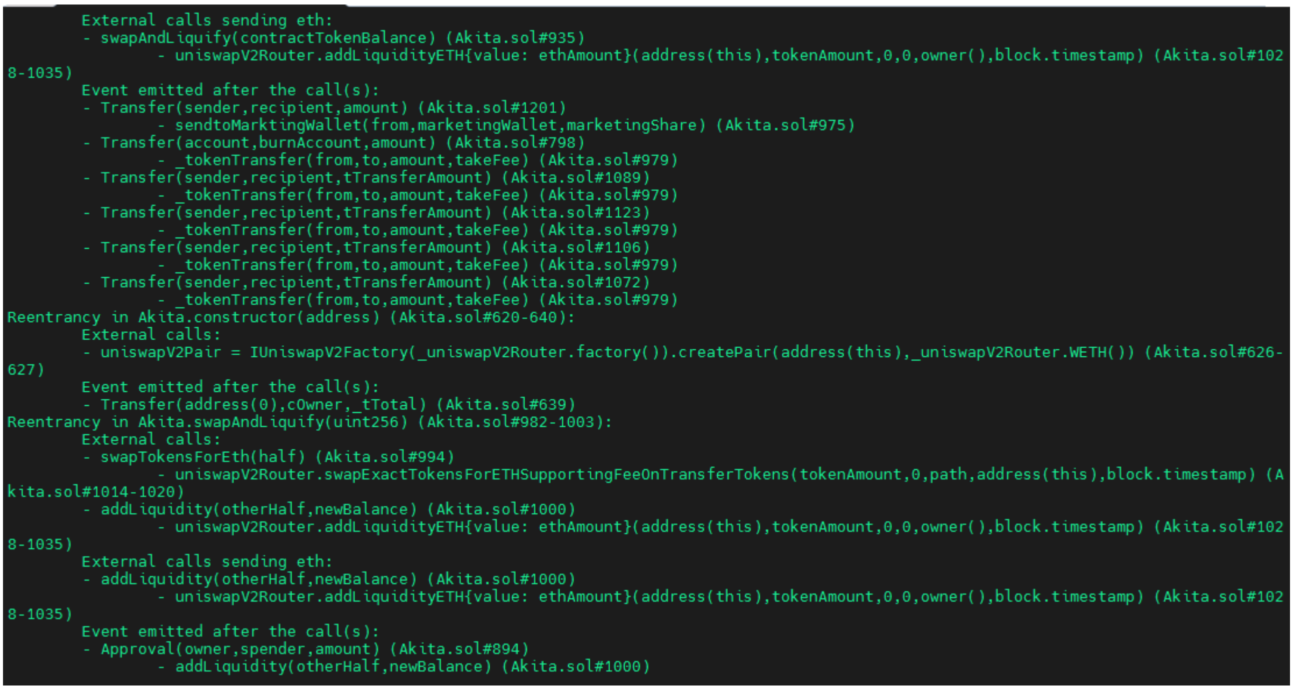
```
        - _liquidityFee = _BuyliquidityFee (Akita.sol#953)
        - removeAllFee() (Akita.sol#959)
                - _liquidityFee = 0 (Akita.sol#873)
        - _liquidityFee = _SellliquidityFee (Akita.sol#961)
        - _liquidityFee = _transferLiquidityFee (Akita.sol#968)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _liquidityFee = _previousLiquidityFee (Akita.sol#880)
                - _liquidityFee = 0 (Akita.sol#873)
        - removeAllFee() (Akita.sol#951)
                - _marketingFee = 0 (Akita.sol#875)
        - _marketingFee = _BuymarketingFee (Akita.sol#955)
        - removeAllFee() (Akita.sol#959)
                - _marketingFee = 0 (Akita.sol#875)
        - _marketingFee = _SellmarketingFee (Akita.sol#963)
        - _marketingFee = _transfermarketingFee (Akita.sol#970)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _marketingFee = _previousMarketingFee (Akita.sol#882)
                - _marketingFee = 0 (Akita.sol#875)
        - removeAllFee() (Akita.sol#951)
                - _previousBurnFee = _burnFee (Akita.sol#870)
        - removeAllFee() (Akita.sol#959)
                - _previousBurnFee = _burnFee (Akita.sol#870)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _previousBurnFee = _burnFee (Akita.sol#870)
        - removeAllFee() (Akita.sol#951)
                - _previousLiquidityFee = _liquidityFee (Akita.sol#869)
        - removeAllFee() (Akita.sol#959)
                - _previousLiquidityFee = _liquidityFee (Akita.sol#869)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _previousLiquidityFee = _liquidityFee (Akita.sol#869)
        - removeAllFee() (Akita.sol#951)
                - _previousMarketingFee = _marketingFee (Akita.sol#871)
        - removeAllFee() (Akita.sol#959)
                - _previousMarketingFee = _marketingFee (Akita.sol#871)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _previousMarketingFee = _marketingFee (Akita.sol#871)
        - removeAllFee() (Akita.sol#951)
                - _previousTaxFee = _taxFee (Akita.sol#868)
```



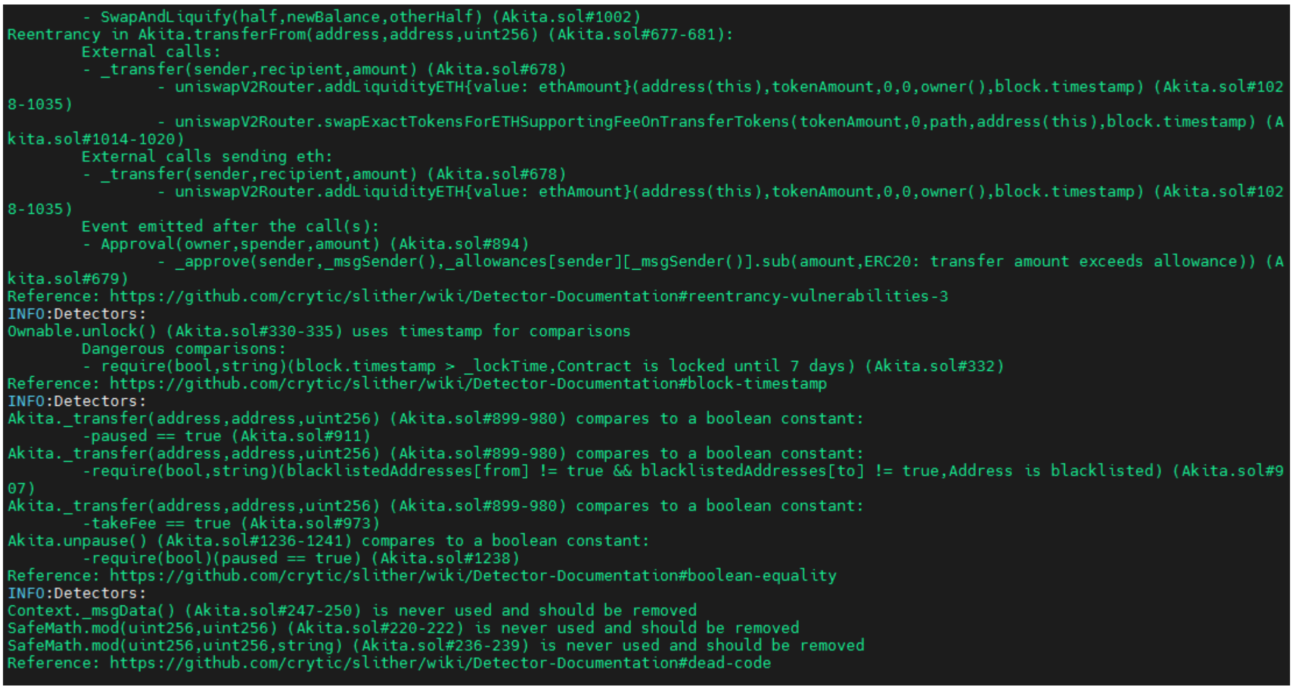
```
        - removeAllFee() (Akita.sol#959)
                - _previousTaxFee = _taxFee (Akita.sol#868)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _previousTaxFee = _taxFee (Akita.sol#868)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _tBurnTotal = _tBurnTotal.add(tBurn) (Akita.sol#1131)
                - _tBurnTotal += amount (Akita.sol#796)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _tFeeTotal = _tFeeTotal.add(tFee) (Akita.sol#1130)
        - removeAllFee() (Akita.sol#951)
                - _taxFee = 0 (Akita.sol#872)
        - _taxFee = _BuytaxFee (Akita.sol#952)
        - removeAllFee() (Akita.sol#959)
                - _taxFee = 0 (Akita.sol#872)
        - _taxFee = _SelltaxFee (Akita.sol#960)
        - _taxFee = _transferTaxFee (Akita.sol#967)
        - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
                - _taxFee = _previousTaxFee (Akita.sol#879)
                - _taxFee = 0 (Akita.sol#872)
        - marketingShare = calculateMarketingFee(amount) (Akita.sol#974)
Reentrancy in Akita.constructor(address) (Akita.sol#620-640):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Akita.sol#626-
627)
        State variables written after the call(s):
        - _isExcludedFromAutoLiquidity[uniswapV2Pair] = true (Akita.sol#636)
        - _isExcludedFromAutoLiquidity[address(_uniswapV2Router)] = true (Akita.sol#637)
        - _isExcludedFromFee[owner()] = true (Akita.sol#633)
        - _isExcludedFromFee[address(this)] = true (Akita.sol#634)
        - uniswapV2Router = _uniswapV2Router (Akita.sol#630)
Reentrancy in Akita.swapAndLiquify(uint256) (Akita.sol#982-1003):
        External calls:
        - swapTokensForEth(half) (Akita.sol#994)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
```



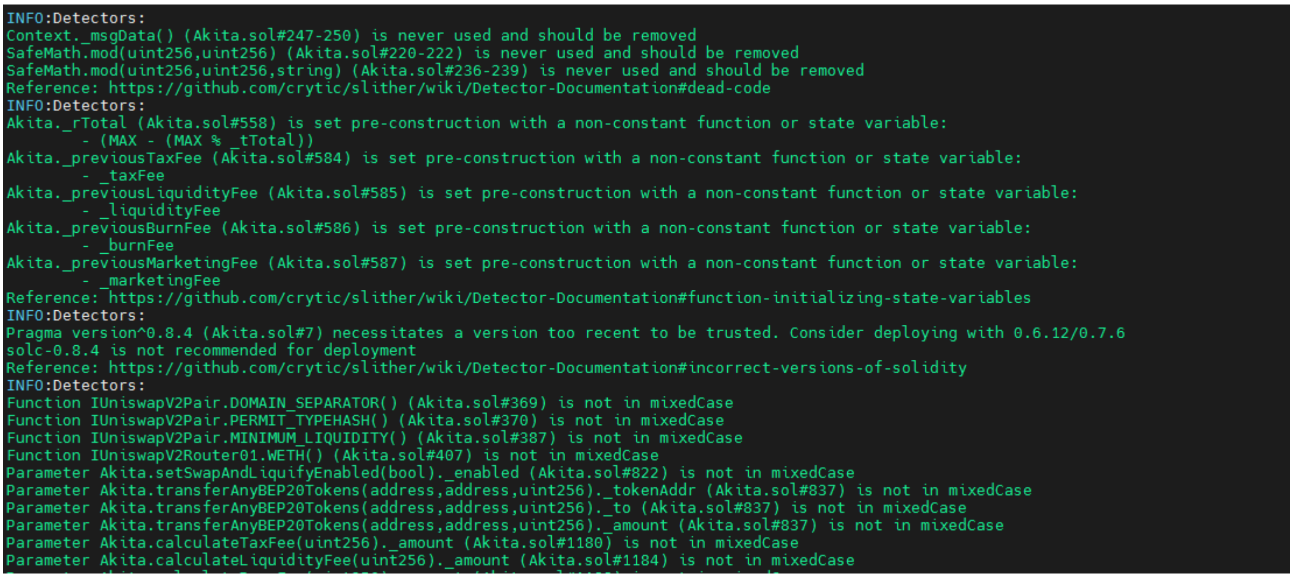
```
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
                - _allowances[owner][spender] = amount (Akita.sol#893)
Reentrancy in Akita.transferFrom(address,address,uint256) (Akita.sol#677-681):
        External calls:
        - _transfer(sender,recipient,amount) (Akita.sol#678)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Akita.sol#678)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (Akita.sol
#679)
                - _allowances[owner][spender] = amount (Akita.sol#893)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Akita._transfer(address,address,uint256) (Akita.sol#899-980):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
```



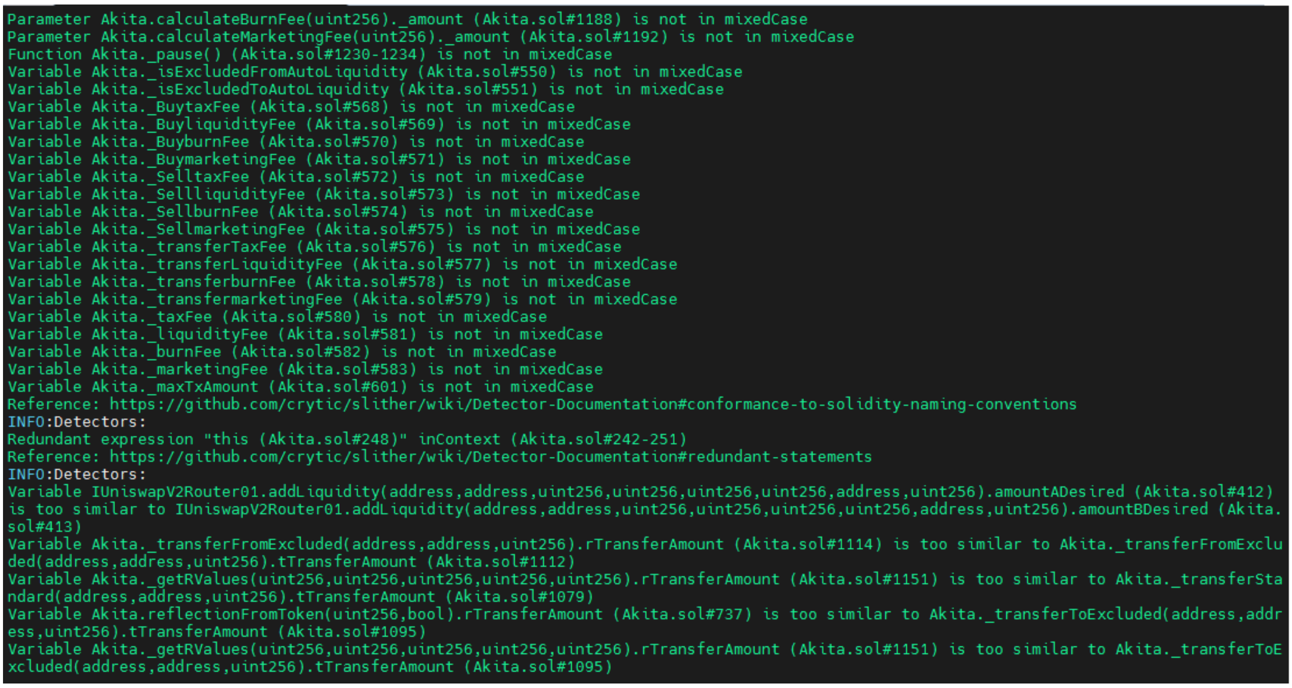
```
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Akita.sol#935)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (Akita.sol#1201)
                - sendtoMarktingWallet(from,marketingWallet,marketingShare) (Akita.sol#975)
        - Transfer(account,burnAccount,amount) (Akita.sol#798)
                - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
        - Transfer(sender,recipient,tTransferAmount) (Akita.sol#1089)
                - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
        - Transfer(sender,recipient,tTransferAmount) (Akita.sol#1123)
                - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
        - Transfer(sender,recipient,tTransferAmount) (Akita.sol#1106)
                - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
        - Transfer(sender,recipient,tTransferAmount) (Akita.sol#1072)
                - _tokenTransfer(from,to,amount,takeFee) (Akita.sol#979)
Reentrancy in Akita.constructor(address) (Akita.sol#620-640):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Akita.sol#626-
627)
        Event emitted after the call(s):
        - Transfer(address(0),cOwner,_tTotal) (Akita.sol#639)
Reentrancy in Akita.swapAndLiquify(uint256) (Akita.sol#982-1003):
        External calls:
        - swapTokensForEth(half) (Akita.sol#994)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Akita.sol#894)
                - addLiquidity(otherHalf,newBalance) (Akita.sol#1000)
```



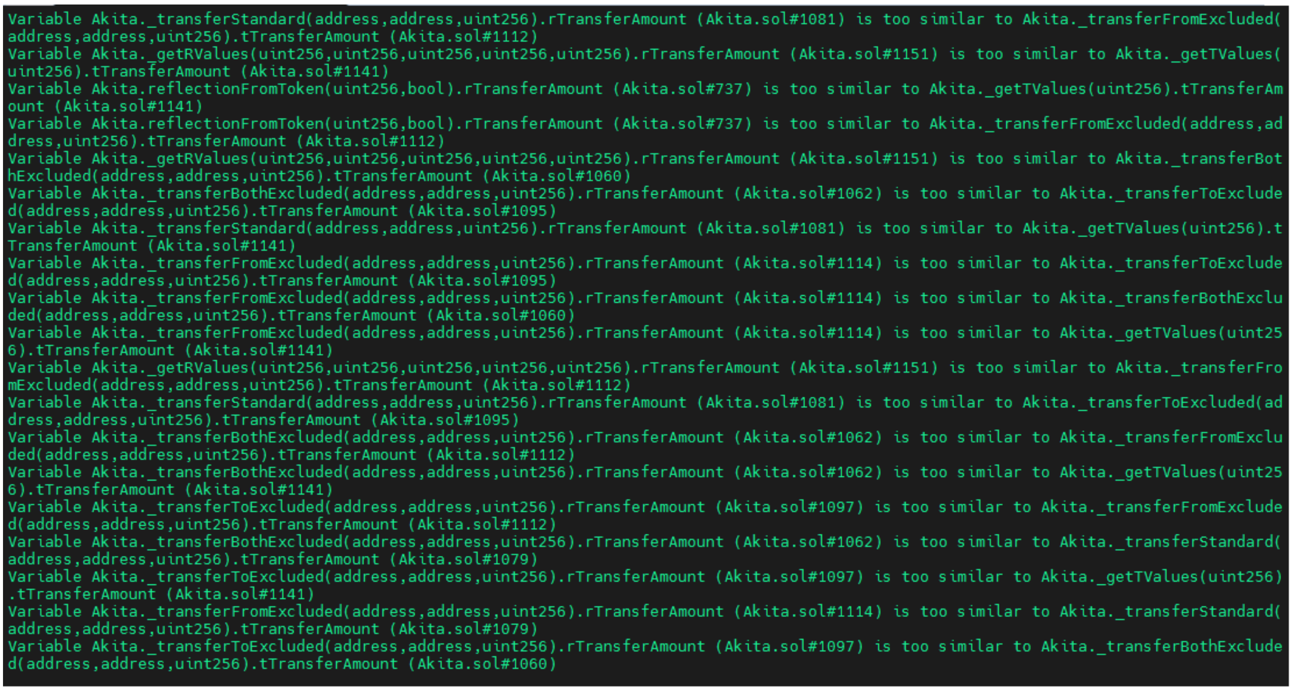
```
        - SwapAndLiquify(half,newBalance,otherHalf) (Akita.sol#1002)
Reentrancy in Akita.transferFrom(address,address,uint256) (Akita.sol#677-681):
        External calls:
        - _transfer(sender,recipient,amount) (Akita.sol#678)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (A
kita.sol#1014-1020)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Akita.sol#678)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Akita.sol#102
8-1035)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Akita.sol#894)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (A
kita.sol#679)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (Akita.sol#330-335) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (Akita.sol#332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Akita._transfer(address,address,uint256) (Akita.sol#899-980) compares to a boolean constant:
        -paused == true (Akita.sol#911)
Akita._transfer(address,address,uint256) (Akita.sol#899-980) compares to a boolean constant:
        -require(bool,string)(blacklistedAddresses[from] != true && blacklistedAddresses[to] != true,Address is blacklisted) (Akita.sol#9
07)
Akita._transfer(address,address,uint256) (Akita.sol#899-980) compares to a boolean constant:
        -takeFee == true (Akita.sol#973)
Akita.unpause() (Akita.sol#1236-1241) compares to a boolean constant:
        -require(bool)(paused == true) (Akita.sol#1238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (Akita.sol#247-250) is never used and should be removed
SafeMath.mod(uint256,uint256) (Akita.sol#220-222) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Akita.sol#236-239) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



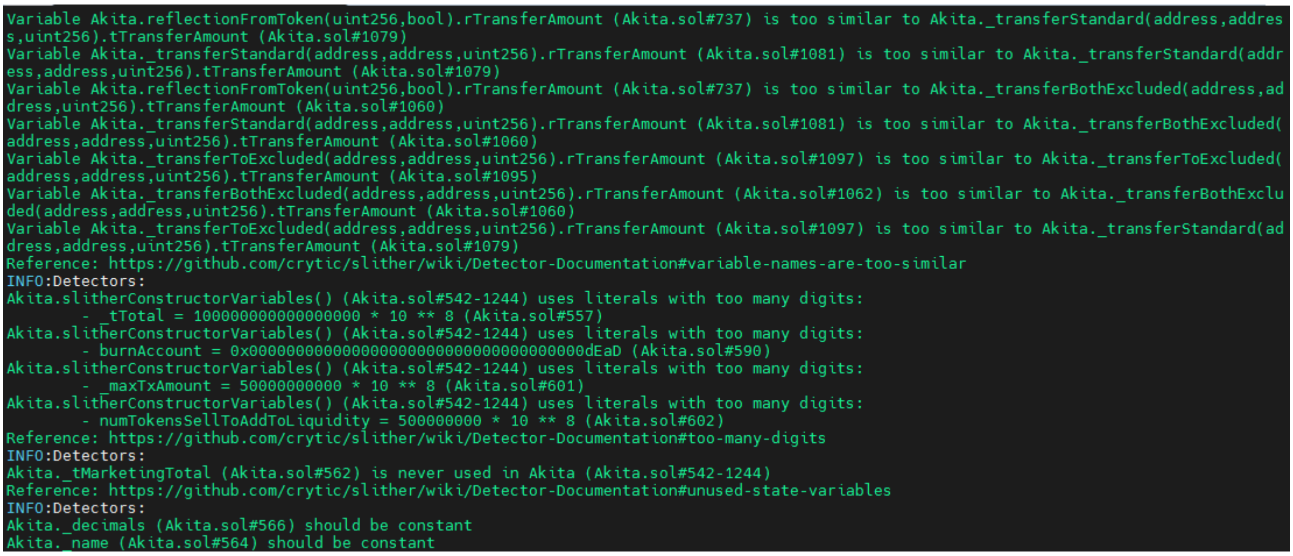
```
INFO:Detectors:
Context._msgData() (Akita.sol#247-250) is never used and should be removed
SafeMath.mod(uint256,uint256) (Akita.sol#220-222) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Akita.sol#236-239) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Akita._rTotal (Akita.sol#558) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Akita._previousTaxFee (Akita.sol#584) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Akita._previousLiquidityFee (Akita.sol#585) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Akita._previousBurnFee (Akita.sol#586) is set pre-construction with a non-constant function or state variable:
        - _burnFee
Akita._previousMarketingFee (Akita.sol#587) is set pre-construction with a non-constant function or state variable:
        - _marketingFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.4 (Akita.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Akita.sol#369) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Akita.sol#370) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Akita.sol#387) is not in mixedCase
Function IUniswapV2Router01.WETH() (Akita.sol#407) is not in mixedCase
Parameter Akita.setSwapAndLiquifyEnabled(bool)._enabled (Akita.sol#822) is not in mixedCase
Parameter Akita.transferAnyBEP20Tokens(address,address,uint256)._tokenAddr (Akita.sol#837) is not in mixedCase
Parameter Akita.transferAnyBEP20Tokens(address,address,uint256)._to (Akita.sol#837) is not in mixedCase
Parameter Akita.transferAnyBEP20Tokens(address,address,uint256)._amount (Akita.sol#837) is not in mixedCase
Parameter Akita.calculateTaxFee(uint256)._amount (Akita.sol#1180) is not in mixedCase
Parameter Akita.calculateLiquidityFee(uint256)._amount (Akita.sol#1184) is not in mixedCase
```



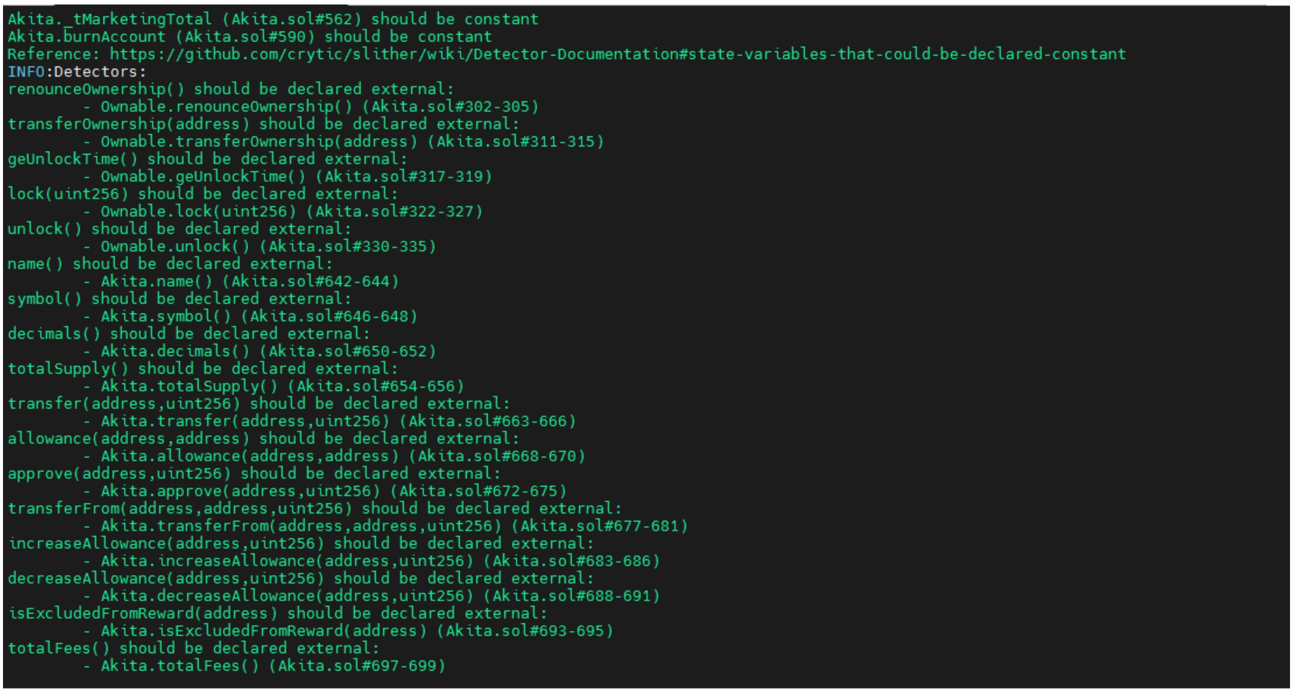
```
Parameter Akita.calculateBurnFee(uint256)._amount (Akita.sol#1188) is not in mixedCase
Parameter Akita.calculateMarketingFee(uint256)._amount (Akita.sol#1192) is not in mixedCase
Function Akita._pause() (Akita.sol#1230-1234) is not in mixedCase
Variable Akita._isExcludedFromAutoLiquidity (Akita.sol#550) is not in mixedCase
Variable Akita._isExcludedToAutoLiquidity (Akita.sol#551) is not in mixedCase
Variable Akita._BuytaxFee (Akita.sol#568) is not in mixedCase
Variable Akita._BuyliquidityFee (Akita.sol#569) is not in mixedCase
Variable Akita._BuyburnFee (Akita.sol#570) is not in mixedCase
Variable Akita._BuymarketingFee (Akita.sol#571) is not in mixedCase
Variable Akita._SelltaxFee (Akita.sol#572) is not in mixedCase
Variable Akita._SellliquidityFee (Akita.sol#573) is not in mixedCase
Variable Akita._SellburnFee (Akita.sol#574) is not in mixedCase
Variable Akita._SellmarketingFee (Akita.sol#575) is not in mixedCase
Variable Akita._transferTaxFee (Akita.sol#576) is not in mixedCase
Variable Akita._transferLiquidityFee (Akita.sol#577) is not in mixedCase
Variable Akita._transferburnFee (Akita.sol#578) is not in mixedCase
Variable Akita._transfermarketingFee (Akita.sol#579) is not in mixedCase
Variable Akita._taxFee (Akita.sol#580) is not in mixedCase
Variable Akita._liquidityFee (Akita.sol#581) is not in mixedCase
Variable Akita._burnFee (Akita.sol#582) is not in mixedCase
Variable Akita._marketingFee (Akita.sol#583) is not in mixedCase
Variable Akita._maxTxAmount (Akita.sol#601) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Akita.sol#248)" inContext (Akita.sol#242-251)
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 (Akita.sol#412)
is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Akita.
sol#413)
Variable Akita._transferFromExcluded(address,address,uint256).rTransferAmount (Akita.sol#1114) is too similar to Akita._transferFromExclu
ded(address,address,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Akita.sol#1151) is too similar to Akita._transferSta
ndard(address,address,uint256).tTransferAmount (Akita.sol#1079)
Variable Akita.reflectionFromToken(uint256,bool).rTransferAmount (Akita.sol#737) is too similar to Akita._transferToExcluded(address,addr
ess,uint256).tTransferAmount (Akita.sol#1095)
Variable Akita._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Akita.sol#1151) is too similar to Akita._transferToE
xcluded(address,address,uint256).tTransferAmount (Akita.sol#1095)
```



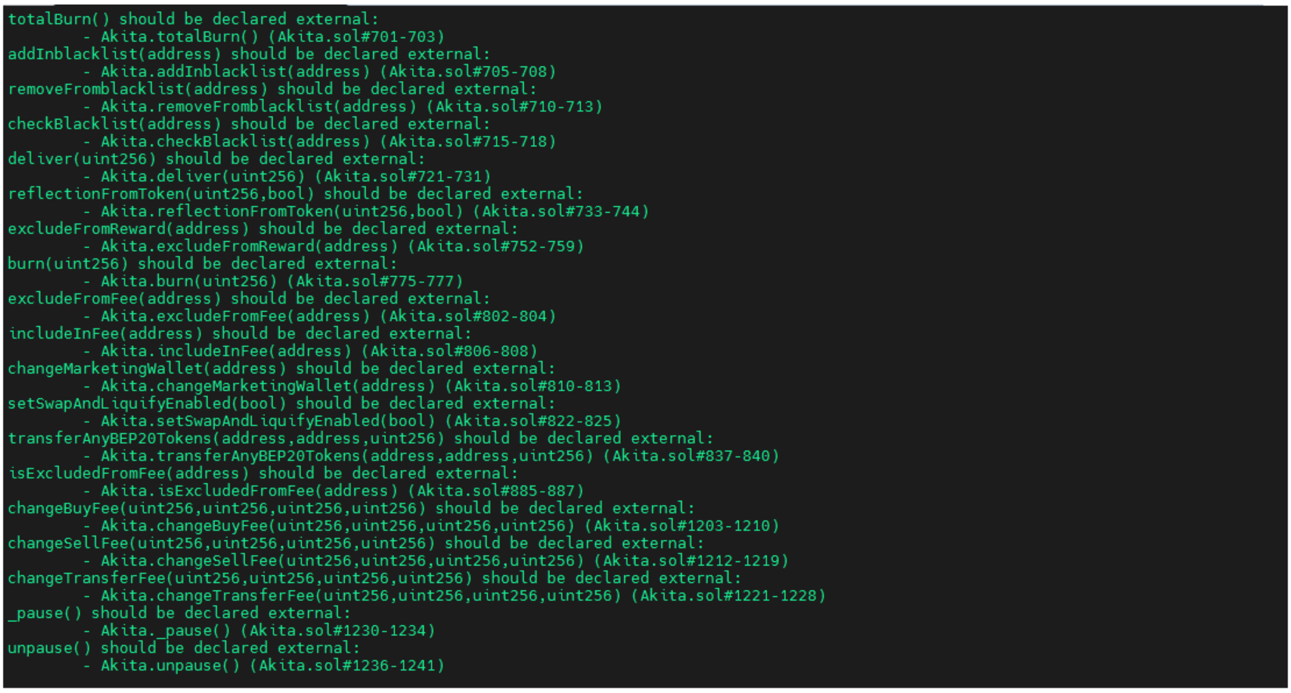
```
Variable Akita._transferStandard(address,address,uint256).rTransferAmount (Akita.sol#1081) is too similar to Akita._transferFromExcluded(
address,address,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Akita.sol#1151) is too similar to Akita._getTValues(
uint256).tTransferAmount (Akita.sol#1141)
Variable Akita.reflectionFromToken(uint256,bool).rTransferAmount (Akita.sol#737) is too similar to Akita._getTValues(uint256).tTransferAm
ount (Akita.sol#1141)
Variable Akita.reflectionFromToken(uint256,bool).rTransferAmount (Akita.sol#737) is too similar to Akita._transferFromExcluded(address,ad
dress,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Akita.sol#1151) is too similar to Akita._transferBot
hExcluded(address,address,uint256).tTransferAmount (Akita.sol#1060)
Variable Akita._transferBothExcluded(address,address,uint256).rTransferAmount (Akita.sol#1062) is too similar to Akita._transferToExclude
d(address,address,uint256).tTransferAmount (Akita.sol#1095)
Variable Akita._transferStandard(address,address,uint256).rTransferAmount (Akita.sol#1081) is too similar to Akita._getTValues(uint256).t
TransferAmount (Akita.sol#1141)
Variable Akita._transferFromExcluded(address,address,uint256).rTransferAmount (Akita.sol#1114) is too similar to Akita._transferToExclude
d(address,address,uint256).tTransferAmount (Akita.sol#1095)
Variable Akita._transferFromExcluded(address,address,uint256).rTransferAmount (Akita.sol#1114) is too similar to Akita._transferBothExclu
ded(address,address,uint256).tTransferAmount (Akita.sol#1060)
Variable Akita._transferFromExcluded(address,address,uint256).rTransferAmount (Akita.sol#1114) is too similar to Akita._getTValues(uint25
6).tTransferAmount (Akita.sol#1141)
Variable Akita._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (Akita.sol#1151) is too similar to Akita._transferFro
mExcluded(address,address,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._transferStandard(address,address,uint256).rTransferAmount (Akita.sol#1081) is too similar to Akita._transferToExcluded(ad
dress,address,uint256).tTransferAmount (Akita.sol#1095)
Variable Akita._transferBothExcluded(address,address,uint256).rTransferAmount (Akita.sol#1062) is too similar to Akita._transferFromExclu
ded(address,address,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._transferBothExcluded(address,address,uint256).rTransferAmount (Akita.sol#1062) is too similar to Akita._getTValues(uint25
6).tTransferAmount (Akita.sol#1141)
Variable Akita._transferToExcluded(address,address,uint256).rTransferAmount (Akita.sol#1097) is too similar to Akita._transferFromExclude
d(address,address,uint256).tTransferAmount (Akita.sol#1112)
Variable Akita._transferBothExcluded(address,address,uint256).rTransferAmount (Akita.sol#1062) is too similar to Akita._transferStandard(
address,address,uint256).tTransferAmount (Akita.sol#1079)
Variable Akita._transferToExcluded(address,address,uint256).rTransferAmount (Akita.sol#1097) is too similar to Akita._getTValues(uint256)
.tTransferAmount (Akita.sol#1141)
Variable Akita._transferFromExcluded(address,address,uint256).rTransferAmount (Akita.sol#1114) is too similar to Akita._transferStandard(
address,address,uint256).tTransferAmount (Akita.sol#1079)
Variable Akita._transferToExcluded(address,address,uint256).rTransferAmount (Akita.sol#1097) is too similar to Akita._transferBothExclude
d(address,address,uint256).tTransferAmount (Akita.sol#1060)
```



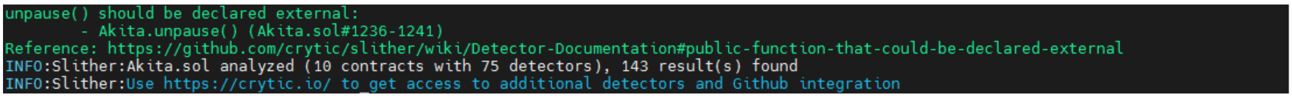
```
Variable Akita.reflectionFromToken(uint256,bool).rTransferAmount (Akita.sol#737) is too similar to Akita._transferStandard(address,addres
s,uint256).tTransferAmount (Akita.sol#1079)
Variable Akita._transferStandard(address,address,uint256).rTransferAmount (Akita.sol#1081) is too similar to Akita._transferStandard(addr
ess,address,uint256).tTransferAmount (Akita.sol#1079)
Variable Akita.reflectionFromToken(uint256,bool).rTransferAmount (Akita.sol#737) is too similar to Akita._transferBothExcluded(address,ad
dress,uint256).tTransferAmount (Akita.sol#1060)
Variable Akita._transferStandard(address,address,uint256).rTransferAmount (Akita.sol#1081) is too similar to Akita._transferBothExcluded(
address,address,uint256).tTransferAmount (Akita.sol#1060)
Variable Akita._transferToExcluded(address,address,uint256).rTransferAmount (Akita.sol#1097) is too similar to Akita._transferToExcluded(
address,address,uint256).tTransferAmount (Akita.sol#1095)
Variable Akita._transferBothExcluded(address,address,uint256).rTransferAmount (Akita.sol#1062) is too similar to Akita._transferBothExclu
ded(address,address,uint256).tTransferAmount (Akita.sol#1060)
Variable Akita._transferToExcluded(address,address,uint256).rTransferAmount (Akita.sol#1097) is too similar to Akita._transferStandard(ad
dress,address,uint256).tTransferAmount (Akita.sol#1079)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Akita.slitherConstructorVariables() (Akita.sol#542-1244) uses literals with too many digits:
        - _tTotal = 1000000000000000000 * 10 ** 8 (Akita.sol#557)
Akita.slitherConstructorVariables() (Akita.sol#542-1244) uses literals with too many digits:
        - burnAccount = 0x000000000000000000000000000000000000dEaD (Akita.sol#590)
Akita.slitherConstructorVariables() (Akita.sol#542-1244) uses literals with too many digits:
        - _maxTxAmount = 50000000000 * 10 ** 8 (Akita.sol#601)
Akita.slitherConstructorVariables() (Akita.sol#542-1244) uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 500000000 * 10 ** 8 (Akita.sol#602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Akita._tMarketingTotal (Akita.sol#562) is never used in Akita (Akita.sol#542-1244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
Akita._decimals (Akita.sol#566) should be constant
Akita._name (Akita.sol#564) should be constant
```



```
Akita._tMarketingTotal (Akita.sol#562) should be constant
Akita.burnAccount (Akita.sol#590) 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() (Akita.sol#302-305)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Akita.sol#311-315)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (Akita.sol#317-319)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Akita.sol#322-327)
unlock() should be declared external:
        - Ownable.unlock() (Akita.sol#330-335)
name() should be declared external:
        - Akita.name() (Akita.sol#642-644)
symbol() should be declared external:
        - Akita.symbol() (Akita.sol#646-648)
decimals() should be declared external:
        - Akita.decimals() (Akita.sol#650-652)
totalSupply() should be declared external:
        - Akita.totalSupply() (Akita.sol#654-656)
transfer(address,uint256) should be declared external:
        - Akita.transfer(address,uint256) (Akita.sol#663-666)
allowance(address,address) should be declared external:
        - Akita.allowance(address,address) (Akita.sol#668-670)
approve(address,uint256) should be declared external:
        - Akita.approve(address,uint256) (Akita.sol#672-675)
transferFrom(address,address,uint256) should be declared external:
        - Akita.transferFrom(address,address,uint256) (Akita.sol#677-681)
increaseAllowance(address,uint256) should be declared external:
        - Akita.increaseAllowance(address,uint256) (Akita.sol#683-686)
decreaseAllowance(address,uint256) should be declared external:
        - Akita.decreaseAllowance(address,uint256) (Akita.sol#688-691)
isExcludedFromReward(address) should be declared external:
        - Akita.isExcludedFromReward(address) (Akita.sol#693-695)
totalFees() should be declared external:
        - Akita.totalFees() (Akita.sol#697-699)
```



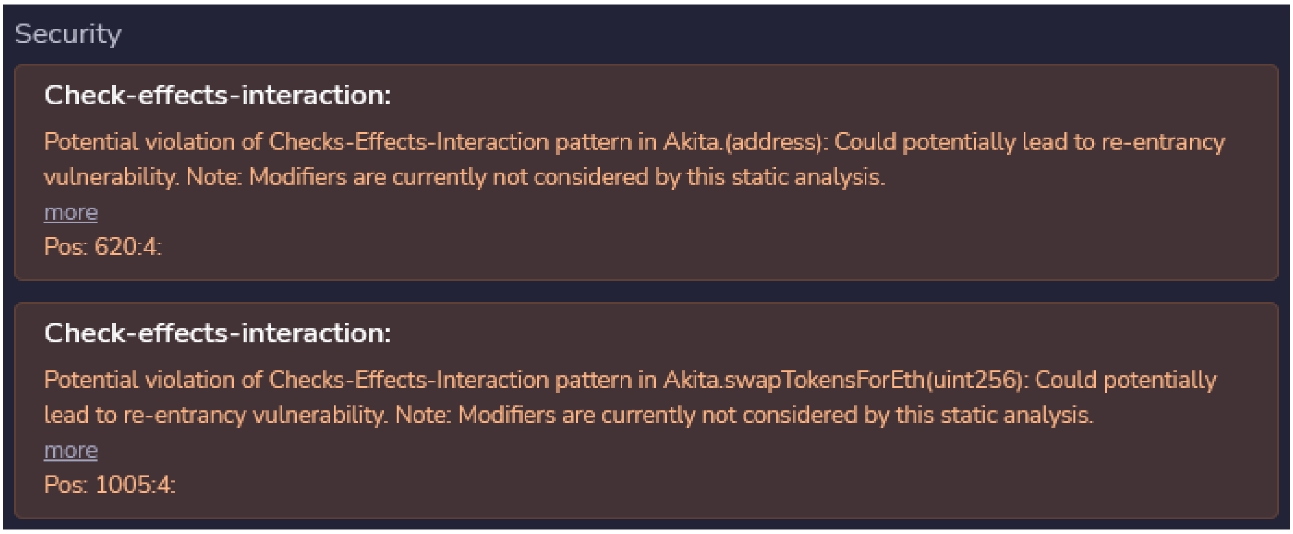
```
totalBurn() should be declared external:
        - Akita.totalBurn() (Akita.sol#701-703)
addInblacklist(address) should be declared external:
        - Akita.addInblacklist(address) (Akita.sol#705-708)
removeFromblacklist(address) should be declared external:
        - Akita.removeFromblacklist(address) (Akita.sol#710-713)
checkBlacklist(address) should be declared external:
        - Akita.checkBlacklist(address) (Akita.sol#715-718)
deliver(uint256) should be declared external:
        - Akita.deliver(uint256) (Akita.sol#721-731)
reflectionFromToken(uint256,bool) should be declared external:
        - Akita.reflectionFromToken(uint256,bool) (Akita.sol#733-744)
excludeFromReward(address) should be declared external:
        - Akita.excludeFromReward(address) (Akita.sol#752-759)
burn(uint256) should be declared external:
        - Akita.burn(uint256) (Akita.sol#775-777)
excludeFromFee(address) should be declared external:
        - Akita.excludeFromFee(address) (Akita.sol#802-804)
includeInFee(address) should be declared external:
        - Akita.includeInFee(address) (Akita.sol#806-808)
changeMarketingWallet(address) should be declared external:
        - Akita.changeMarketingWallet(address) (Akita.sol#810-813)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - Akita.setSwapAndLiquifyEnabled(bool) (Akita.sol#822-825)
transferAnyBEP20Tokens(address,address,uint256) should be declared external:
        - Akita.transferAnyBEP20Tokens(address,address,uint256) (Akita.sol#837-840)
isExcludedFromFee(address) should be declared external:
        - Akita.isExcludedFromFee(address) (Akita.sol#885-887)
changeBuyFee(uint256,uint256,uint256,uint256) should be declared external:
        - Akita.changeBuyFee(uint256,uint256,uint256,uint256) (Akita.sol#1203-1210)
changeSellFee(uint256,uint256,uint256,uint256) should be declared external:
        - Akita.changeSellFee(uint256,uint256,uint256,uint256) (Akita.sol#1212-1219)
changeTransferFee(uint256,uint256,uint256,uint256) should be declared external:
        - Akita.changeTransferFee(uint256,uint256,uint256,uint256) (Akita.sol#1221-1228)
_pause() should be declared external:
        - Akita._pause() (Akita.sol#1230-1234)
unpause() should be declared external:
        - Akita.unpause() (Akita.sol#1236-1241)
```



```
unpause() should be declared external:
        - Akita.unpause() (Akita.sol#1236-1241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Akita.sol analyzed (10 contracts with 75 detectors), 143 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

shibapy.sol



Security

Check-effects-interaction:

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

more

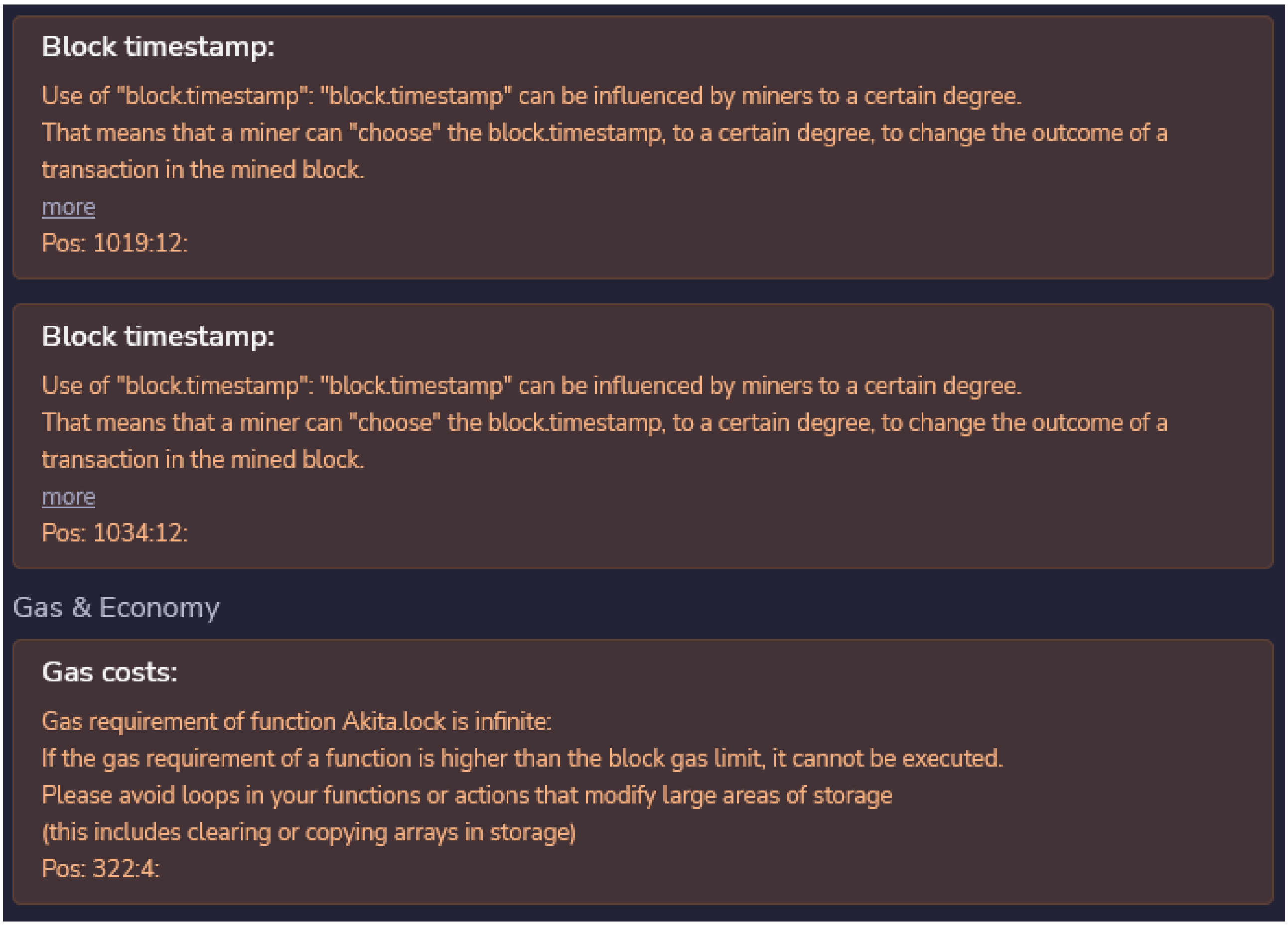
Pos: 620:4:

Check-effects-interaction:

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

more

Pos: 1005:4:



Block timestamp:

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

more

Pos: 1019:12:

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.

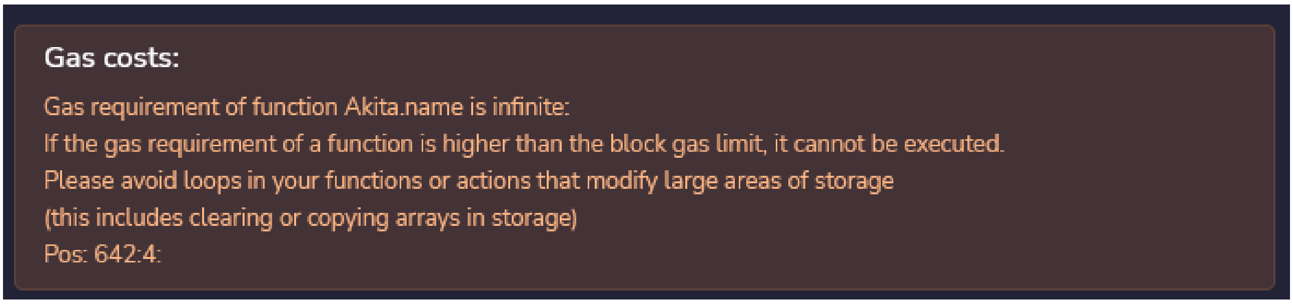
more

Pos: 1034:12:

Gas & Economy

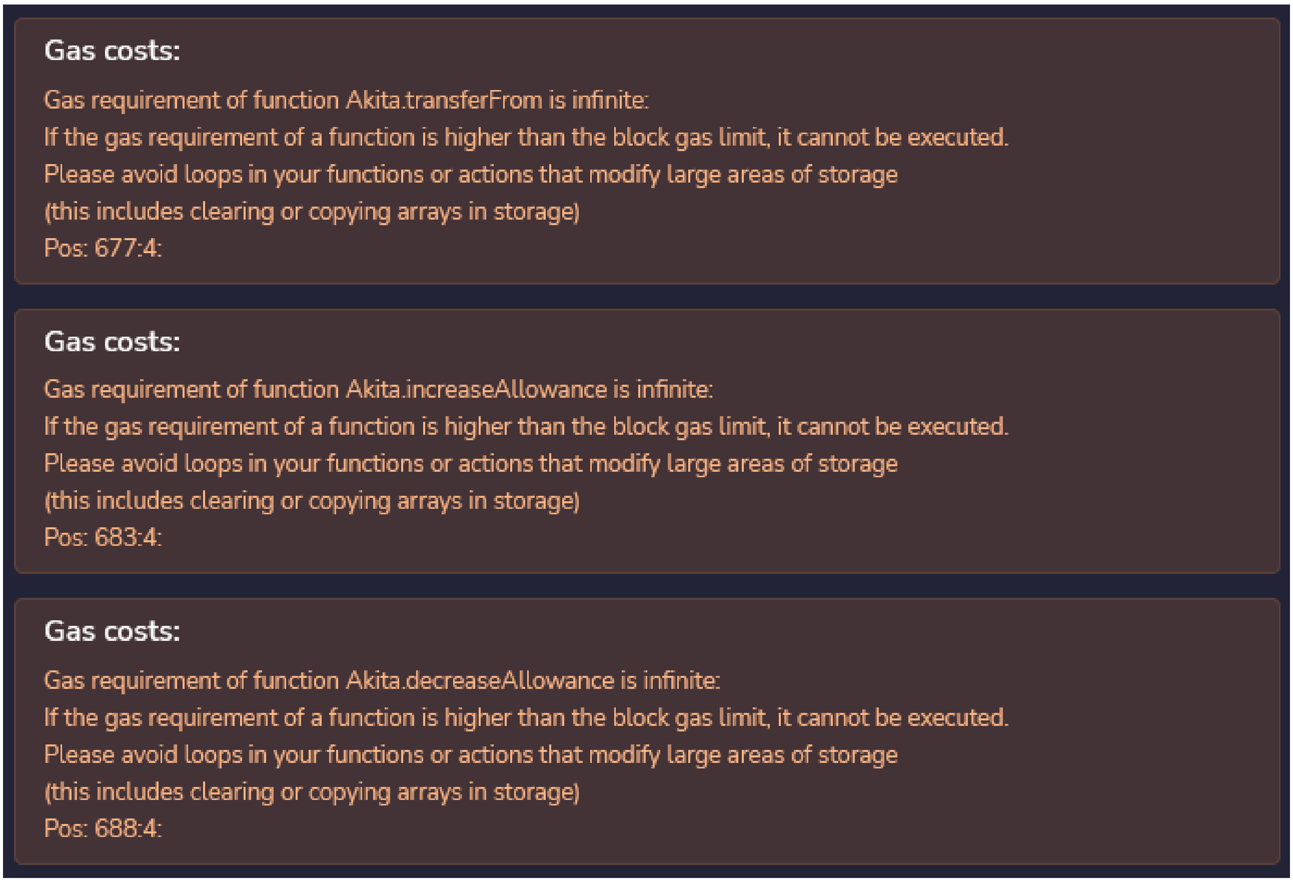
Gas costs:

Gas requirement of function Akita.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: 322:4:



Gas costs:

Gas requirement of function Akita.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: 642:4:



Gas costs:

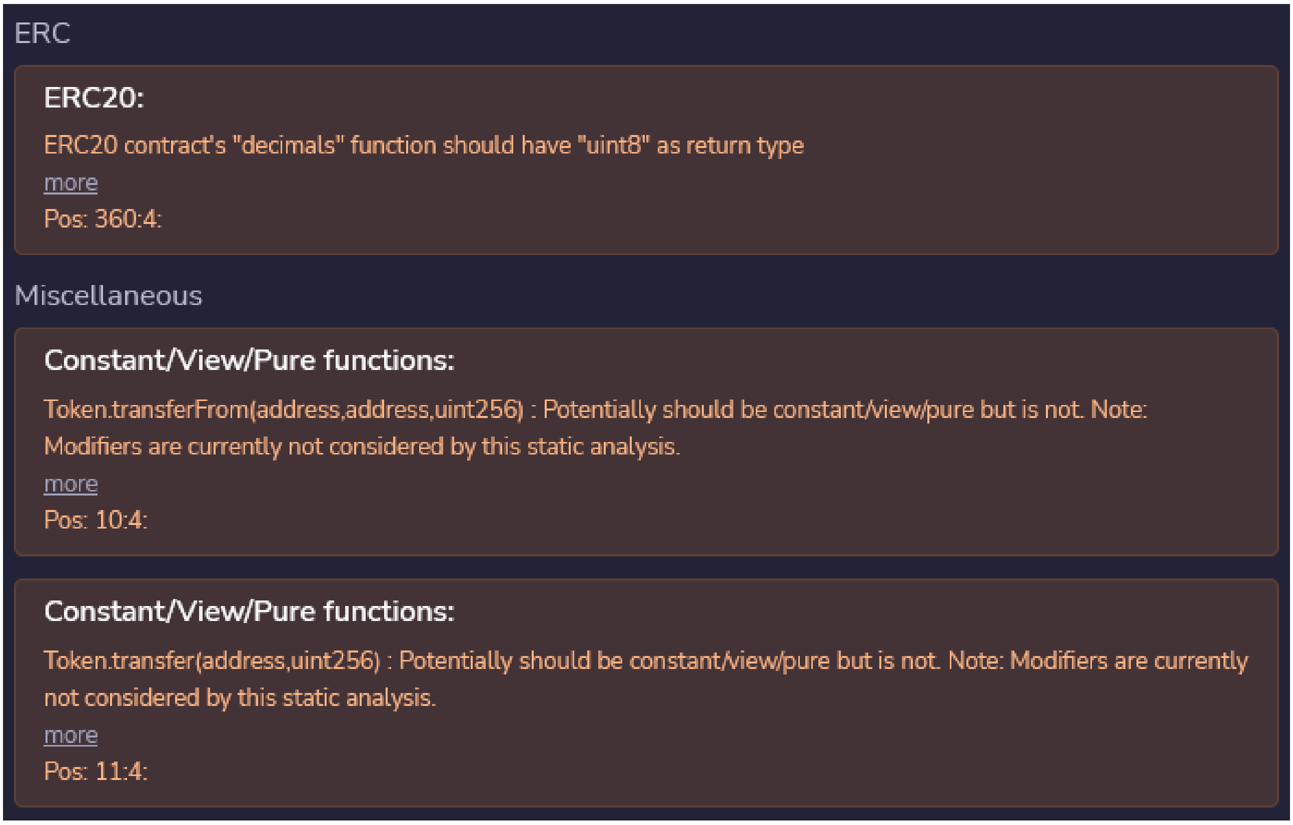
Gas requirement of function Akita.transferFrom is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 677:4:

Gas costs:

Gas requirement of function Akita.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: 683:4:

Gas costs:

Gas requirement of function Akita.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: 688:4:



ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 360:4:

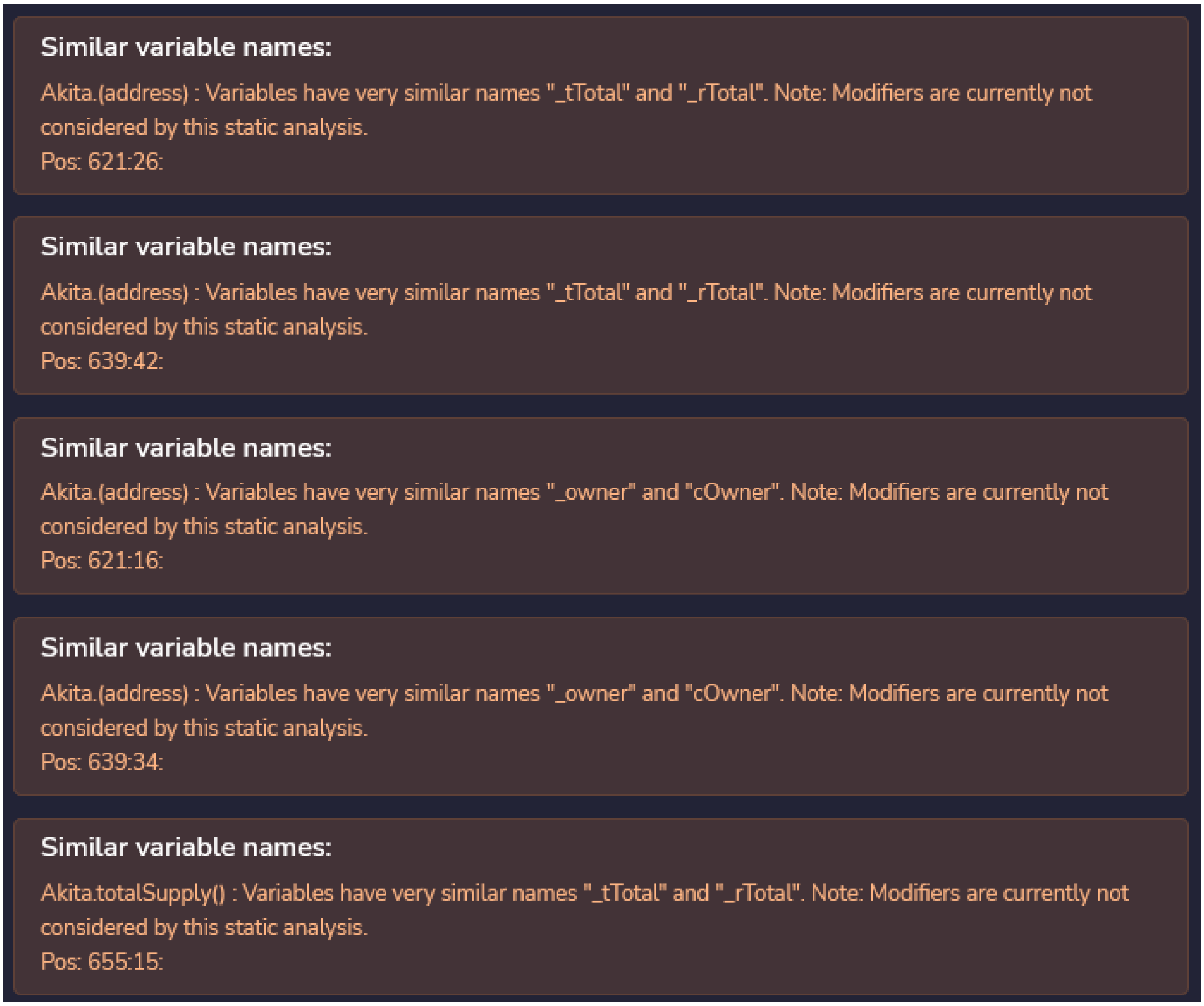
Miscellaneous

Constant/View/Pure functions:

Token.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 10:4:

Constant/View/Pure functions:

Token.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 11:4:



Similar variable names:

Akita.(address) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 621:26:

Similar variable names:

Akita.(address) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 639:42:

Similar variable names:

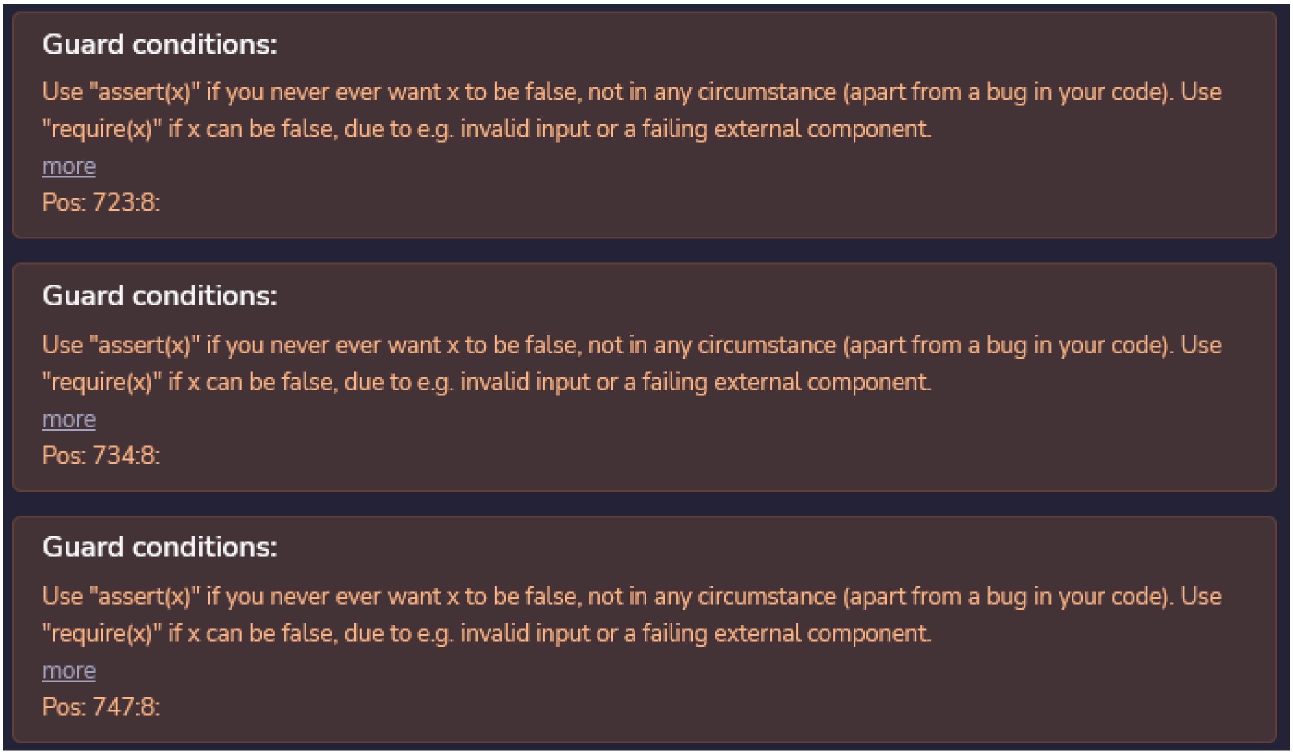
Akita.(address) : Variables have very similar names "_owner" and "cOwner". Note: Modifiers are currently not considered by this static analysis.
Pos: 621:16:

Similar variable names:

Akita.(address) : Variables have very similar names "_owner" and "cOwner". Note: Modifiers are currently not considered by this static analysis.
Pos: 639:34:

Similar variable names:

Akita.totalSupply() : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.
Pos: 655:15:



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: 723:8:

Guard conditions:

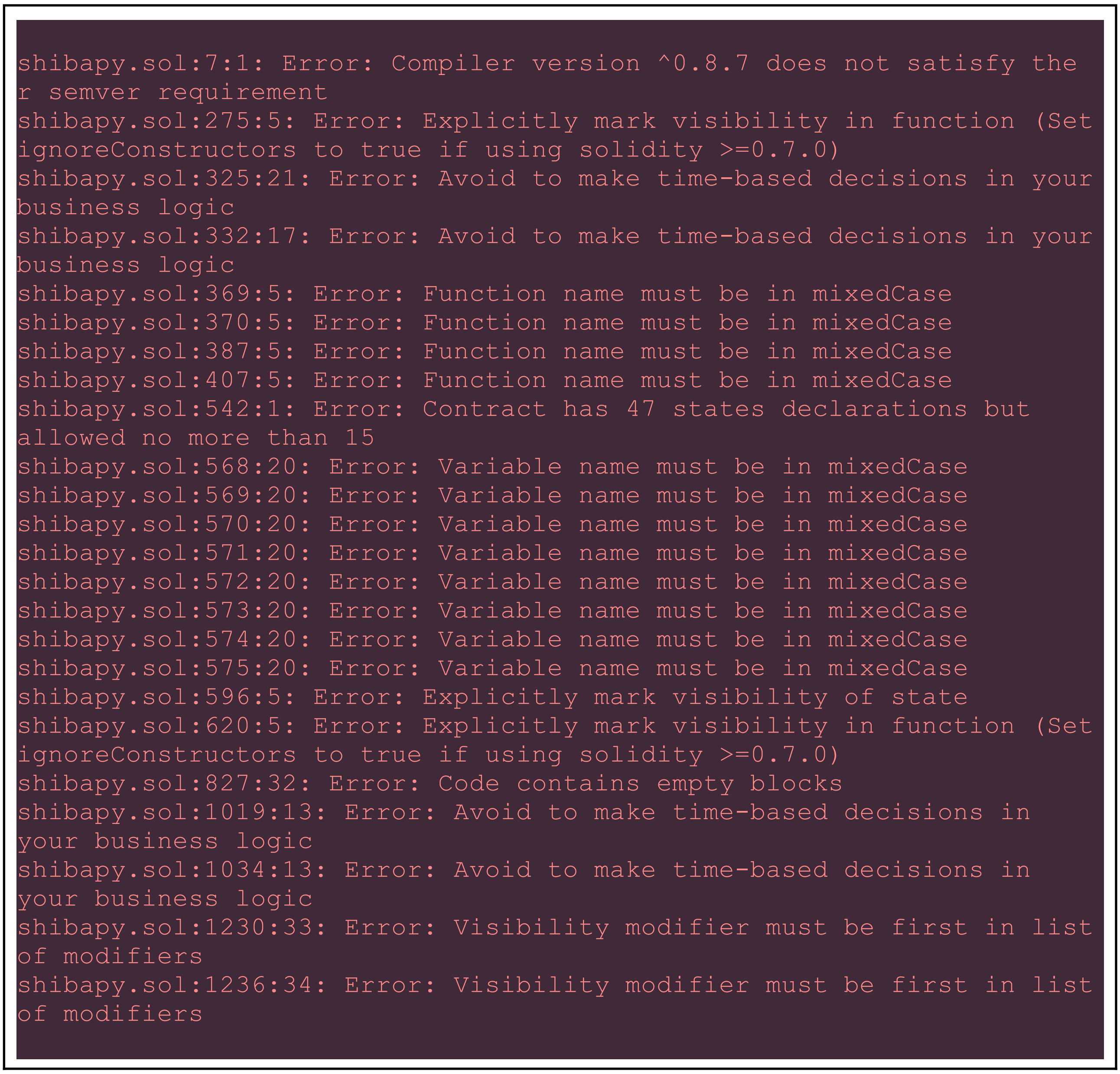
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 734:8:

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: 747:8:

Solhint Linter

shibapy.sol



```
shibapy.sol:7:1: Error: Compiler version ^0.8.7 does not satisfy the
r semver requirement
shibapy.sol:275:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
shibapy.sol:325:21: Error: Avoid to make time-based decisions in your
business logic
shibapy.sol:332:17: Error: Avoid to make time-based decisions in your
business logic
shibapy.sol:369:5: Error: Function name must be in mixedCase
shibapy.sol:370:5: Error: Function name must be in mixedCase
shibapy.sol:387:5: Error: Function name must be in mixedCase
shibapy.sol:407:5: Error: Function name must be in mixedCase
shibapy.sol:542:1: Error: Contract has 47 states declarations but
allowed no more than 15
shibapy.sol:568:20: Error: Variable name must be in mixedCase
shibapy.sol:569:20: Error: Variable name must be in mixedCase
shibapy.sol:570:20: Error: Variable name must be in mixedCase
shibapy.sol:571:20: Error: Variable name must be in mixedCase
shibapy.sol:572:20: Error: Variable name must be in mixedCase
shibapy.sol:573:20: Error: Variable name must be in mixedCase
shibapy.sol:574:20: Error: Variable name must be in mixedCase
shibapy.sol:575:20: Error: Variable name must be in mixedCase
shibapy.sol:596:5: Error: Explicitly mark visibility of state
shibapy.sol:620:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
shibapy.sol:827:32: Error: Code contains empty blocks
shibapy.sol:1019:13: Error: Avoid to make time-based decisions in
your business logic
shibapy.sol:1034:13: Error: Avoid to make time-based decisions in
your business logic
shibapy.sol:1230:33: Error: Visibility modifier must be first in list
of modifiers
shibapy.sol:1236:34: 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.

EtherAuthority

Email: audit@EtherAuthority.io