EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	OneSeed Protocol
Website:	https://oneseed.app
Platform:	Etherscan
Language:	Solidity
Date:	December 4th, 2021

Table of contents

Introduction

EtherAuthority was contracted by the OneSeed team to perform the Security audit of the OneSeed Protocol smart contracts 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 December 4th, 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

The goal of the OneSeed contract is to develop an ERC20-based token. There is a stack component and a referral component to allow users to stake their tokens and receive a return based on the amount in staking. The returns are used to remunerate the team, partners and liquidity.

Audit scope

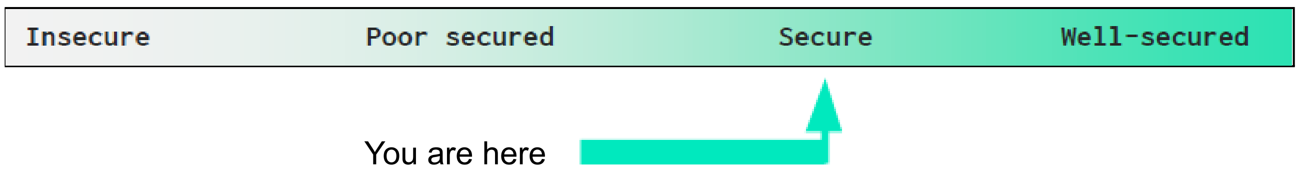
Name	Code Review and Security Analysis Report for OneSeed Protocol Smart Contracts
Platform	**Etherscan / Solidity**
File 1	OneSeed.sol
File 1 MD5 Hash	AA799FE6BBB50D7AF6A24EBC0A9D774C
File 2	OneSeedPublicSale.sol
File 2 MD5 Hash	7F547464209C798F95FC3A94E1E33ECA
Audit Date	December 4th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 OneSeed.sol ● Name: OneSeed ● Symbol: SEED ● Decimals: 2 ● Maximum Supply: 300 Million ● Total Supply: 105 Million ● Liquidity Fee: 5% ● Number Tokens Sell To Add To Liquidity: 5000 ● Interest Modification: 1000 ● Partner Reward: 50000 ● Team Reward: 25000 ● Liquidity Reward: 25000 ● Return Divisor: 1 Million ● Minimum Stake Lock: 10 minutes ● Maximum Staking Amount: 3 Million ● Global Maximum Staking Amount: 80 Million ● Minimum Staking Amount:100	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 2 OneSeedPublicSale.sol ● Maximum Allowance:12 Quadrillion ● Minimum Buying Power: 25 Trillion ● Price Adjustment: 1125 ● Maximum Sale Time: 2629746 seconds	**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 1 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	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	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 2 smart contracts files. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in OneSeed Protocol 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 OneSeed Protocol.

The OneSeed Protocol 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 OneSeed Protocol smart contracts code in the form of a code. 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://oneseed.app which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts 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

OneSeed.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	decimals	external	Passed	No Issue
4	symbol	external	Passed	No Issue
5	name	external	Passed	No Issue
6	totalSupply	external	Passed	No Issue
7	balanceOf	write	Passed	No Issue
8	transfer	external	Passed	No Issue
9	allowance	external	Passed	No Issue
10	approve	external	Passed	No Issue
11	transferFrom	external	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	Unlimited minting	Refer Audit Findings
15	isExcluded	read	Passed	No Issue
16	calculateLiquidityFee	read	Passed	No Issue
17	stakeReferred	external	Critical operation lacks event log	Refer Audit Findings
18	stake	write	Critical operation lacks event log	Refer Audit Findings
19	lockStake	external	Critical operation lacks event log	Refer Audit Findings
20	isStaking	read	Passed	No Issue
21	unstake	write	Critical operation lacks event log	Refer Audit Findings
22	getStakingMember	external	Passed	No Issue
23	isUnstakePermitted	read	Passed	No Issue
24	partnerPayout	external	Missing require an error message	Refer Audit Findings
25	getPartnerBalance	external	Passed	No Issue
26	getRewards	external	access Only Active Staker	No Issue
27	adjustInterest	write	Passed	No Issue
28	swapAndLiquify	write	Passed	No Issue
29	swapTokensForEth	write	Passed	No Issue
30	addLiquidity	write	Passed	No Issue
31	exclude	write	Critical operation lacks event log	Refer Audit Findings

32	include	write	Critical operation lacks event log	Refer Audit Findings
33	setLiquidityFee	external	Critical operation lacks event log, Similar functionality at multiple places	Refer Audit Findings
34	enableTrading	external	Critical operation lacks event log	Refer Audit Findings
35	_transfer	internal	Passed	No Issue
36	_mint	internal	Unlimited minting	Refer Audit Findings
37	_burn	internal	Unused functions	Refer Audit Findings
38	_approve	internal	Passed	No Issue
39	_burnFrom	internal	Unused functions	Refer Audit Findings
40	setLiquidityFeePercent	external	Critical operation lacks event log, Similar functionality at multiple places	Refer Audit Findings
41	setSwapAndLiquifyEnabled	write	access only Permitted	No Issue
42	updateMinimumTokensSellToAddToLiquidity	external	Critical operation lacks event log,Missing require an error message	Refer Audit Findings
43	setGlobalMaxStakingAmount	external	Critical operation lacks event log,Missing require an error message	Refer Audit Findings
44	setMinimumStakingAmount	external	Critical operation lacks event log,Missing require an error message	Refer Audit Findings
45	isAutoCompound	external	Passed	No Issue
46	toggleAutoCompound	external	Critical operation lacks event log	Refer Audit Findings
47	getCountMyPartnerRewards	read	Passed	No Issue
48	getCountMyStakeRewards	read	Passed	No Issue
49	getGlobalStakingAmount	external	Passed	No Issue
50	setTeamAddress	external	Critical operation lacks event log,Missing	Refer Audit Findings

			require an error message	
51	getMyStakeReward	external	Passed	No Issue
52	getMyPartnerReward	external	Passed	No Issue
53	isReferred	read	Passed	No Issue
54	totalLiqudityRewardFees	read	Passed	No Issue
55	receive	external	Passed	No Issue
56	OnlyActiveStaker	modifier	Passed	No Issue
57	StakingActive	modifier	Passed	No Issue
58	lockTheSwap	modifier	Passed	No Issue
59	owner	read	Passed	No Issue
60	onlyOwner	modifier	Passed	No Issue
61	onlyPermitted	modifier	Passed	No Issue
62	restrictedOwner	modifier	Passed	No Issue
63	renounceOwnership	write	access only Owner	No Issue
64	transferOwnership	write	access only Owner	No Issue
65	geUnlockTime	read	Passed	No Issue
66	lock	write	access only Owner	No Issue
67	unlock	write	Passed	No Issue

OneSeedPublicSale.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	buy	write	Passed	No Issue
3	receive	external	Passed	No Issue
4	calcTokens	read	Missing require an error message	Refer Audit Findings
5	getRate	read	Passed	No Issue
6	getStagePrices	external	Missing require an error message	Refer Audit Findings
7	addLiquidity	write	Passed	No Issue
8	setMaxAllowance	external	Missing require an error message	Refer Audit Findings
9	startSale	external	Missing require an error message	Refer Audit Findings
10	considerCompleted	external	Missing require an error message	Refer Audit Findings
11	finishSale	external	Missing require an error message	Refer Audit Findings
12	getRemainingAllowance	external	Passed	No Issue
13	getRemainingTokensForSale	read	Passed	No Issue

14	getAllowance	external	Passed	No Issue
15	owner	read	Passed	No Issue
16	onlyOwner	modifier	Passed	No Issue
17	onlyPermitted	modifier	Passed	No Issue
18	restrictedOwner	modifier	Passed	No Issue
19	renounceOwnership	write	access only Owner	No Issue
20	transferOwnership	write	access only Owner	No Issue
21	geUnlockTime	read	Passed	No Issue
22	lock	write	access only Owner	No Issue
23	unlock	write	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

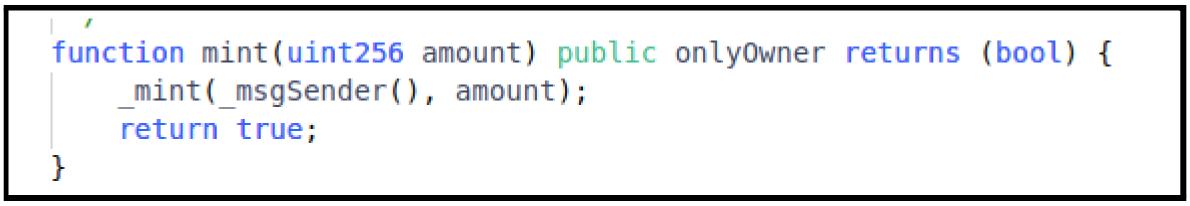
No High severity vulnerabilities were found.

Medium

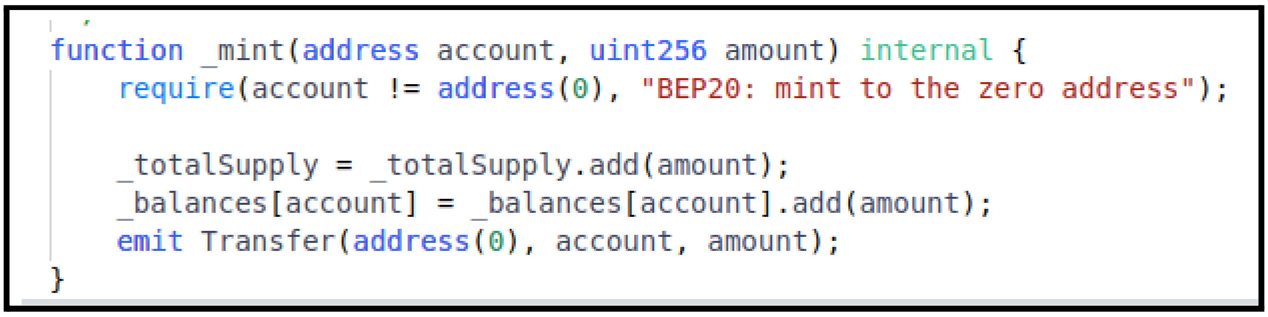
No Medium severity vulnerabilities were found.

Low

(1) Unlimited minting: **- OneSeed.sol**



```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
}
```



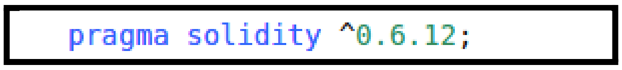
```
function _mint(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: mint to the zero address");

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);
    emit Transfer(address(0), account, amount);
}
```

An 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 disregard this issue.

Very Low / Informational / Best practices:

(1) Use the latest solidity version: **- OneSeed.sol, OneSeedPublicSale.sol**



```
pragma solidity ^0.6.12;
```

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

Resolution: Please use 0.8.10 which is the latest version.

(2) Make variables constant: **- OneSeed.sol, OneSeedPublicSale.sol**

OneSeed.sol

- _decimals
- _symbol
- _name
- _interestModification
- _teamReward
- _liquidityReward
- _returnDivisor
- _rewardTime
- _minimumStakeLock
- _maxStakingAmount

OneSeedPublicSale.sol

- _minBuyingPower
- _maxSaleTime
- _saleFinished
- _initialSellingPrice
- _priceAdjustment

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.

(3) Function input parameters lack of check: **- OneSeed.sol**

Some functions lack validation of the input parameter.

Resolution: We suggest adding validation like

- for integer type variables, check for greater than 0
- for address type variables, check for not address(0)
- for percentage values, check for minimum/maximum range.

  Functions are:
  - mint
  - setLiquidityFeePercent, setLiquidityFee

(4) Unused functions: **- OneSeed.sol**

_burnFrom() and _burn() functions are defined internally but not used anywhere in code.

Resolution: we suggest removing the unused function.

(5) Critical operation lacks event log: **- OneSeed.sol**

There are several places in the smart contracts, where a critical function call event log was not added.

Resolution: We suggest adding logs for functions listed below:

- stakeReferred()
- stake() - StakingActive
- lockStake() - OnlyActiveStaker
- unstake() - OnlyActiveStaker
- exclude() - onlyPermitted
- include() - onlyPermitted
- setLiquidityFee() - onlyPermitted
- enableTrading() - onlyPermitted
- setLiquidityFeePercent() - onlyPermitted
- updateMinimumTokensSellToAddToLiquidity() - onlyPermitted
- setGlobalMaxStakingAmount() - onlyPermitted
- setMinimumStakingAmount() - onlyPermitted
- toggleAutoCompound() - OnlyActiveStaker
- setTeamAddress() - onlyPermitted

(6) Similar functionality at multiple places: **- OneSeed.sol**

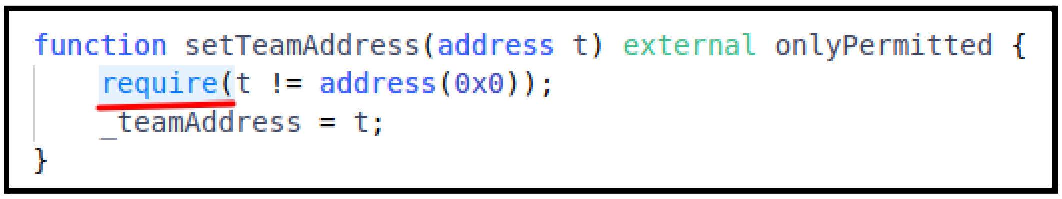
- setLiquidityFee()
- setLiquidityFeePercent()

Both functions are used to set the same variable.

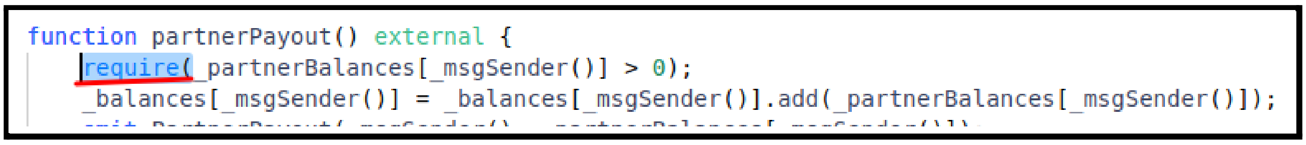
Resolution: We suggest removing the function having logic functionality.

(7) Missing require an error message: **- OneSeed.sol, OneSeedPublicSale.sol**

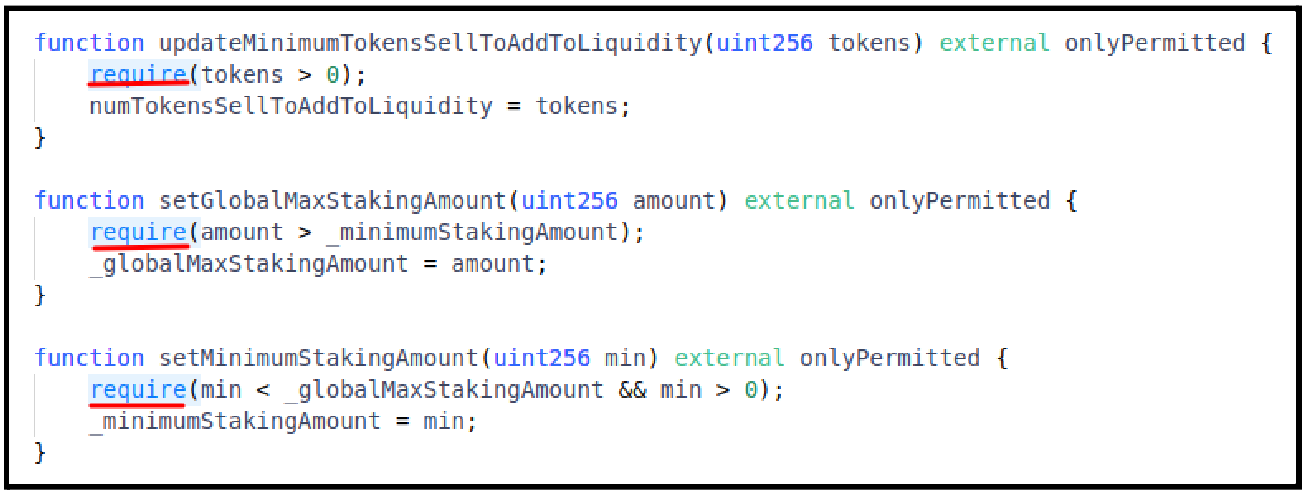
OneSeed.sol



```
function setTeamAddress(address t) external onlyPermitted {
    require(t != address(0x0));
    _teamAddress = t;
}
```

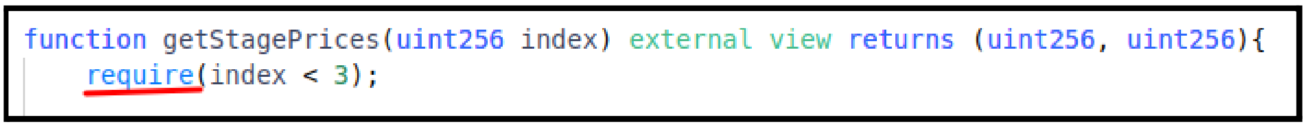


```
function partnerPayout() external {
    require( partnerBalances[_msgSender()] > 0);
    _balances[_msgSender()] = _balances[_msgSender()].add(_partnerBalances[_msgSender()]);
```

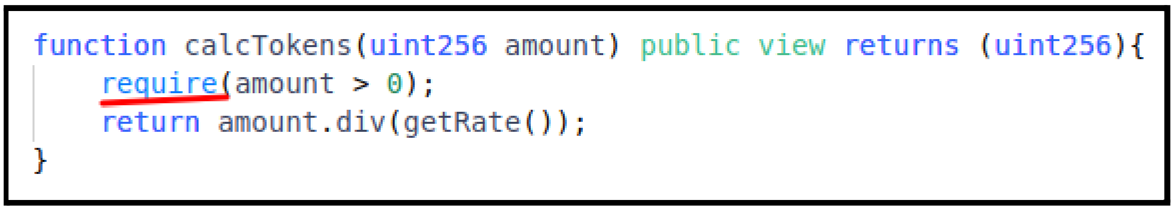


```
function updateMinimumTokensSellToAddToLiquidity(uint256 tokens) external onlyPermitted {
    require(tokens > 0);
    numTokensSellToAddToLiquidity = tokens;
}

function setGlobalMaxStakingAmount(uint256 amount) external onlyPermitted {
    require(amount > _minimumStakingAmount);
    _globalMaxStakingAmount = amount;
}

function setMinimumStakingAmount(uint256 min) external onlyPermitted {
    require(min < _globalMaxStakingAmount && min > 0);
    _minimumStakingAmount = min;
}
```

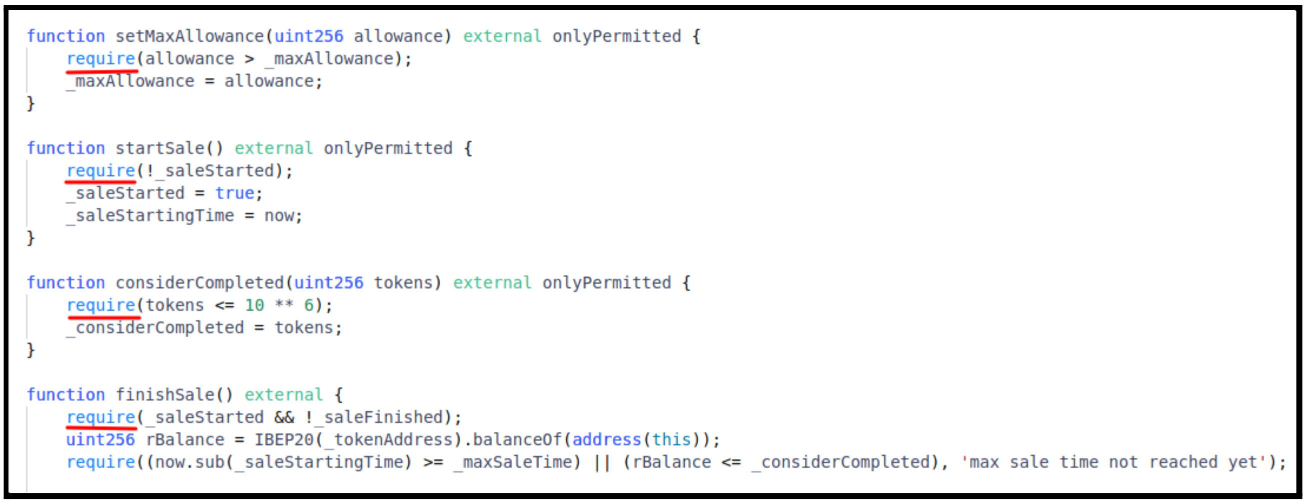
OneSeedPublicSale.sol



```
function getStagePrices(uint256 index) external view returns (uint256, uint256){
    require(index < 3);
```



```
function calcTokens(uint256 amount) public view returns (uint256){
    require(amount > 0);
    return amount.div(getRate());
}
```



```
function setMaxAllowance(uint256 allowance) external onlyPermitted {
    require(allowance > _maxAllowance);
    _maxAllowance = allowance;
}

function startSale() external onlyPermitted {
    require(!_saleStarted);
    _saleStarted = true;
    _saleStartingTime = now;
}

function considerCompleted(uint256 tokens) external onlyPermitted {
    require(tokens <= 10 ** 6);
    _considerCompleted = tokens;
}

function finishSale() external {
    require(_saleStarted && !_saleFinished);
    uint256 rBalance = IBEP20(_tokenAddress).balanceOf(address(this));
    require((now.sub(_saleStartingTime) >= _maxSaleTime) || (rBalance <= _considerCompleted), 'max sale time not reached yet');
```

Require conditions failed and here we don't know to identify why this condition fails in functions.

Resolution: Set proper required error message to identify for failed execution in the function.

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: The OneSeed owner can create `amount` tokens and assigns them to himself, increasing the total supply.
- stake: The OneSeed StakingActive can create new stake.
- lockStake: The OneSeed ActiveStaker can lockstake.
- unstake: The OneSeed ActiveStaker can unstake.
- getRewards: The OneSeed ActiveStaker can get rewards.
- exclude: The OneSeed Permitted owner can exclude status true.
- include: The OneSeed Permitted owner can exclude status false.
- setLiquidityFee: The OneSeed Permitted owner can set a liquidity fee.
- enableTrading: The OneSeed Permitted owner can enable trading true.
- setLiquidityFeePercent: The OneSeed Permitted owner can set liquidity fee percentage.
- setSwapAndLiquifyEnabled: The OneSeed Permitted owner can set liquidity enable.
- updateMinimumTokensSellToAddToLiquidity: The OneSeed Permitted owner can update minimum token sell to add to liquidity.
- setGlobalMaxStakingAmount: The OneSeed Permitted owner can set global maximum staking amount.
- setMinimumStakingAmount: The OneSeed Permitted owner can set a minimum staking amount.
- toggleAutoCompound: The OneSeed ActiveStaker owner can toggle auto compound.
- setTeamAddress: The OneSeed Permitted owner can set a team address.
- setMaxAllowance: The OneSeedPublicSale Permitted owner can set maximum allowance.
- startSale: The OneSeedPublicSale Permitted owner can start sale time.
- considerCompleted: The OneSeedPublicSale Permitted owner can consider completed tokens.

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.

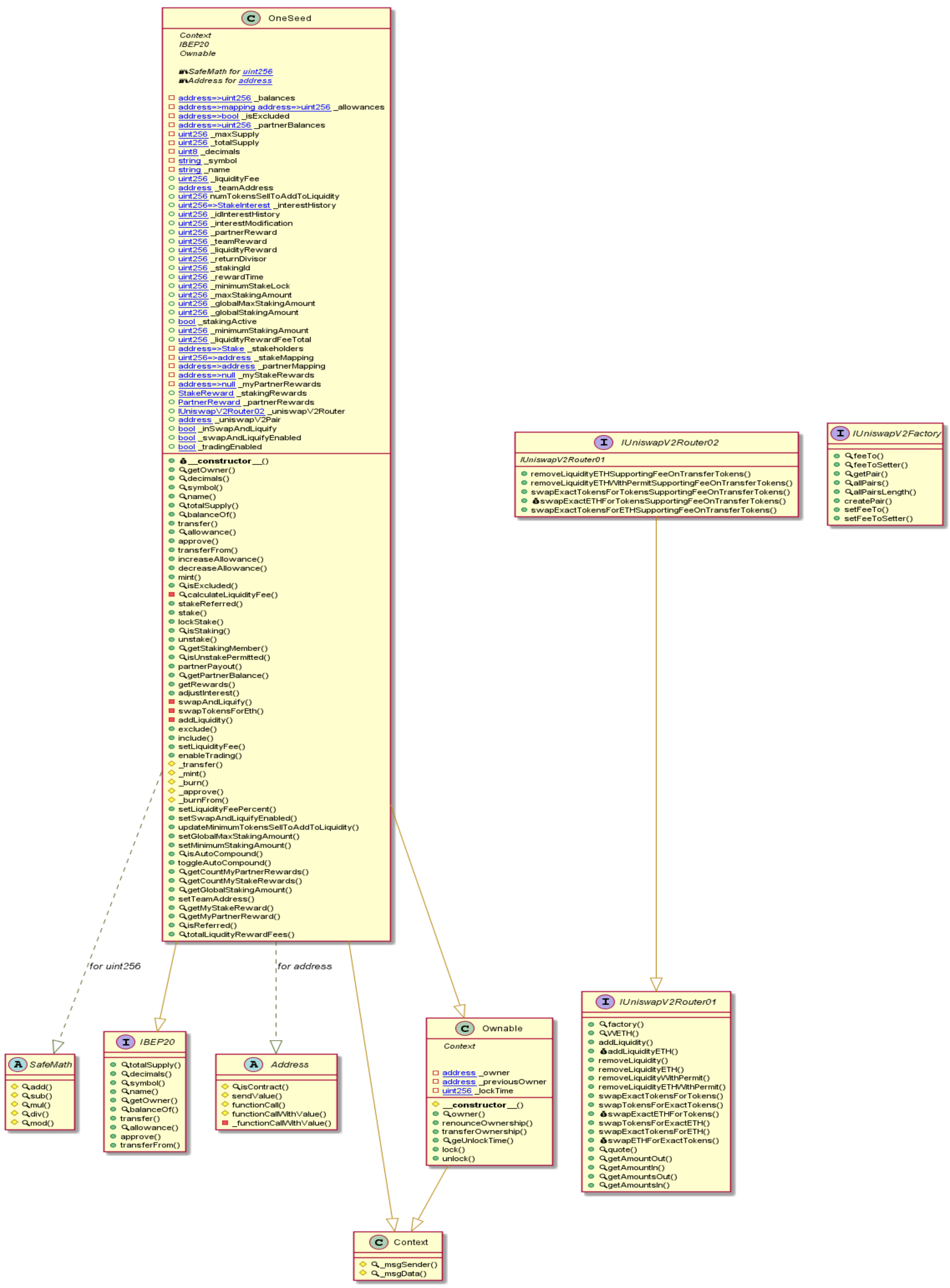
Technical Disclaimer

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.

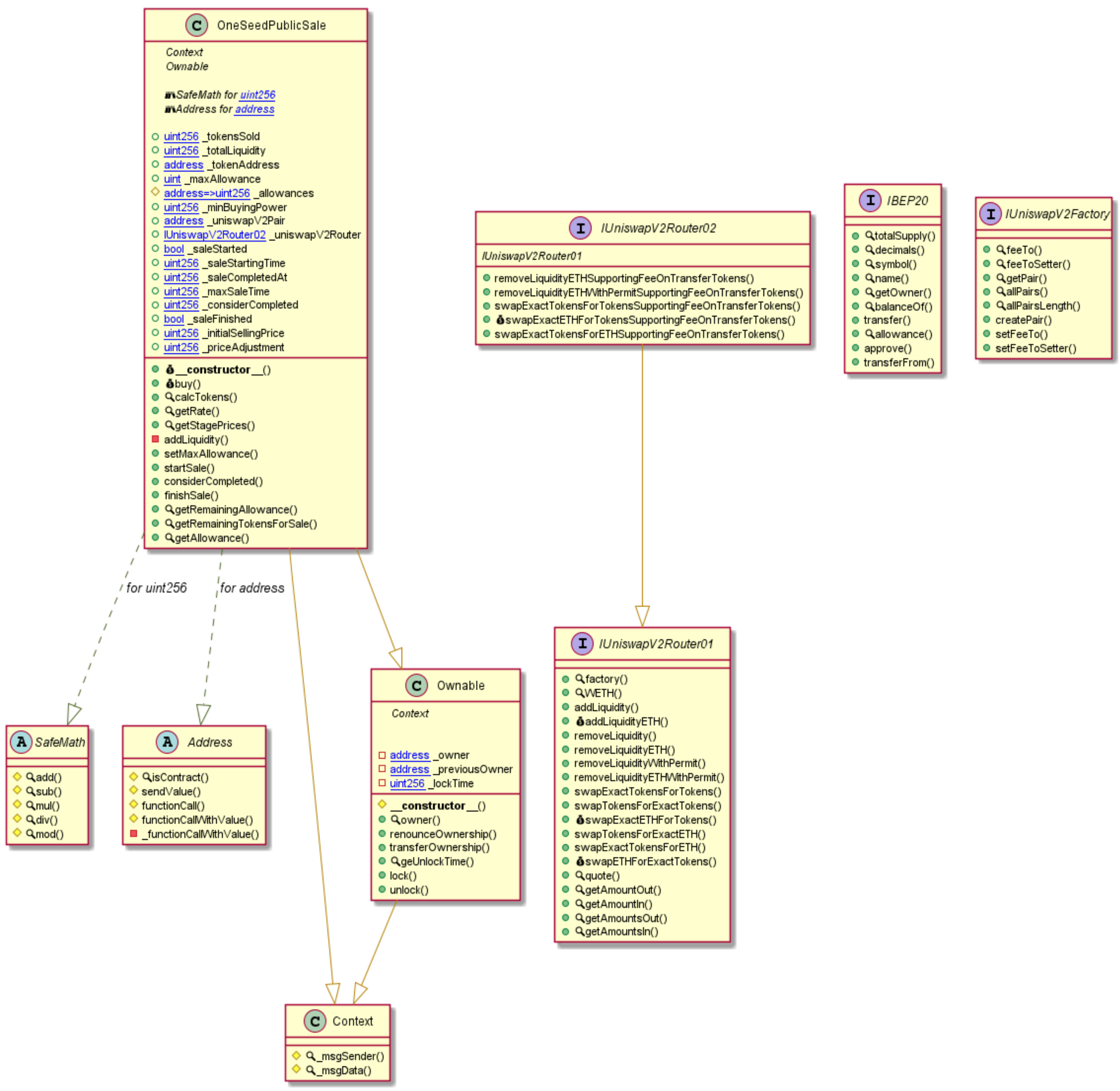
Appendix

Code Flow Diagram - OneSeed Protocol

OneSeed Diagram

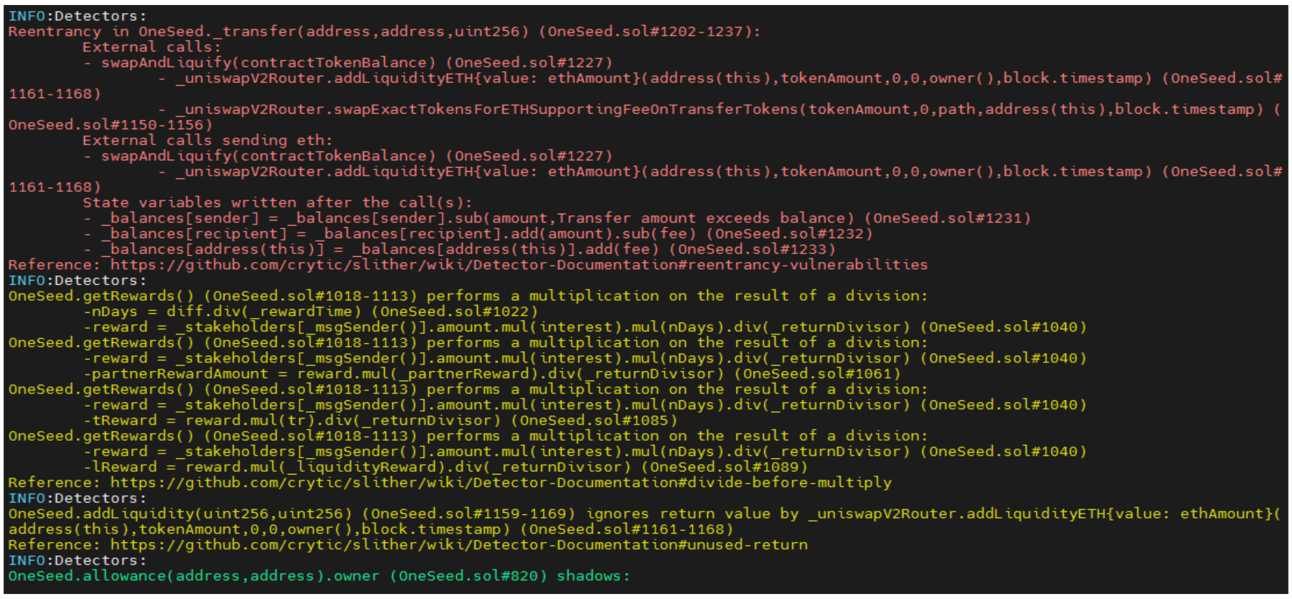


OneSeedPublicSale Diagram

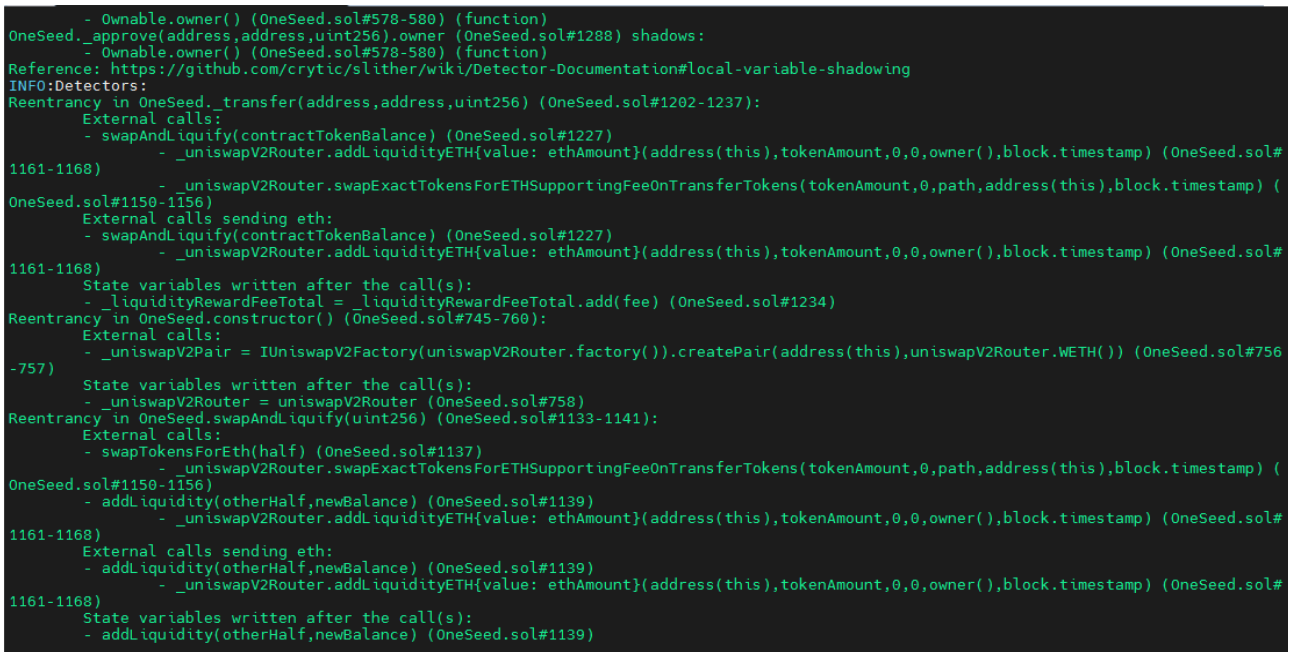


Slither Results Log

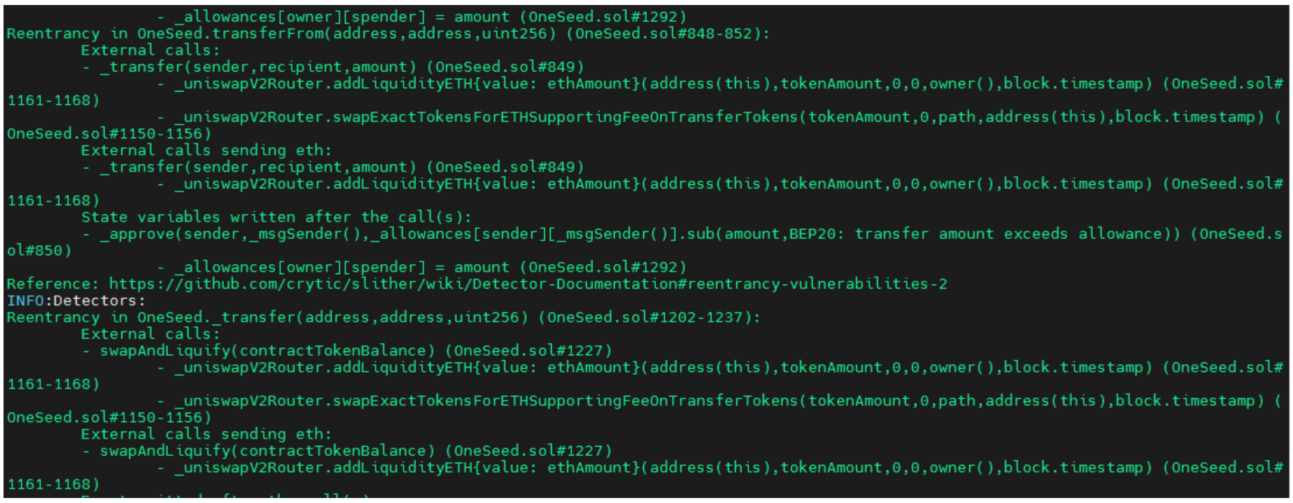
Slither log >> OneSeed.sol



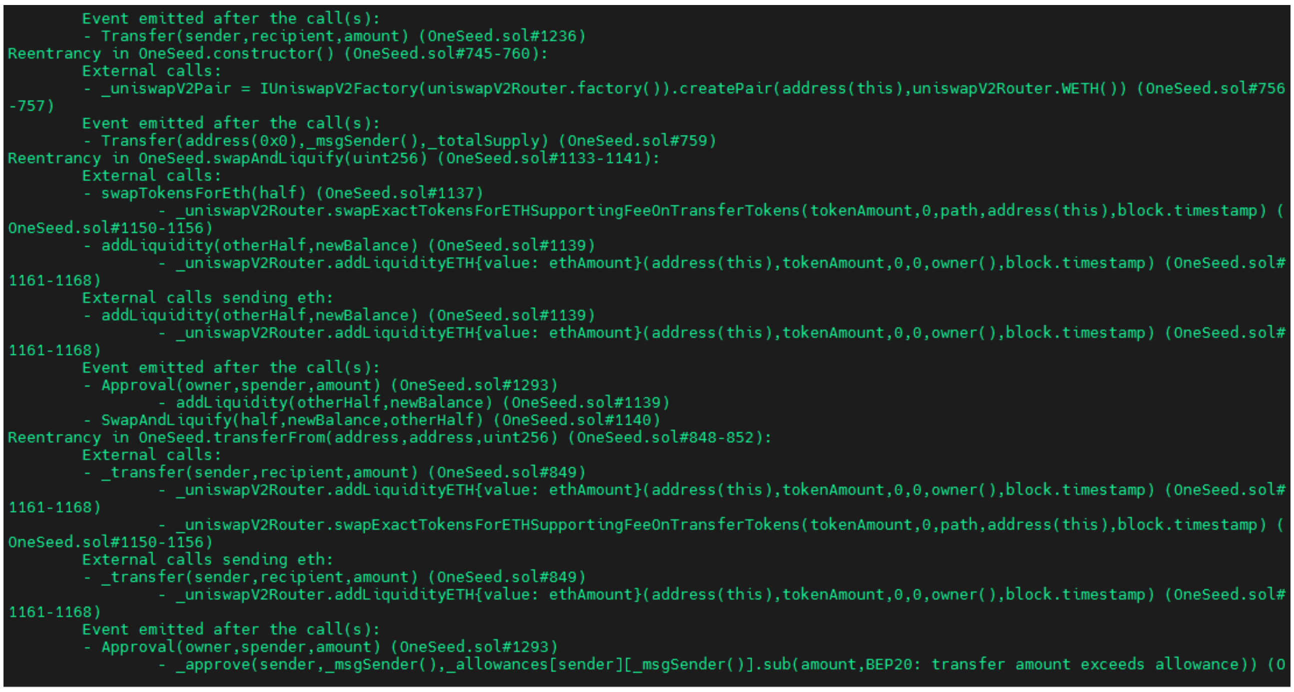
```
INFO:Detectors:
Reentrancy in OneSeed._transfer(address,address,uint256) (OneSeed.sol#1202-1237):
        External calls:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender].sub(amount,Transfer amount exceeds balance) (OneSeed.sol#1231)
        - _balances[recipient] = _balances[recipient].add(amount).sub(fee) (OneSeed.sol#1232)
        - _balances[address(this)] = _balances[address(this)].add(fee) (OneSeed.sol#1233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
OneSeed.getRewards() (OneSeed.sol#1018-1113) performs a multiplication on the result of a division:
        -nDays = diff.div(_rewardTime) (OneSeed.sol#1022)
        -reward = _stakeholders[_msgSender()].amount.mul(interest).mul(nDays).div(_returnDivisor) (OneSeed.sol#1040)
OneSeed.getRewards() (OneSeed.sol#1018-1113) performs a multiplication on the result of a division:
        -reward = _stakeholders[_msgSender()].amount.mul(interest).mul(nDays).div(_returnDivisor) (OneSeed.sol#1040)
        -partnerRewardAmount = reward.mul(_partnerReward).div(_returnDivisor) (OneSeed.sol#1061)
OneSeed.getRewards() (OneSeed.sol#1018-1113) performs a multiplication on the result of a division:
        -reward = _stakeholders[_msgSender()].amount.mul(interest).mul(nDays).div(_returnDivisor) (OneSeed.sol#1040)
        -tReward = reward.mul(tr).div(_returnDivisor) (OneSeed.sol#1085)
OneSeed.getRewards() (OneSeed.sol#1018-1113) performs a multiplication on the result of a division:
        -reward = _stakeholders[_msgSender()].amount.mul(interest).mul(nDays).div(_returnDivisor) (OneSeed.sol#1040)
        -lReward = reward.mul(_liquidityReward).div(_returnDivisor) (OneSeed.sol#1089)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
OneSeed.addLiquidity(uint256,uint256) (OneSeed.sol#1159-1169) ignores return value by _uniswapV2Router.addLiquidityETH{value: ethAmount}(
address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#1161-1168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
OneSeed.allowance(address,address).owner (OneSeed.sol#820) shadows:
```



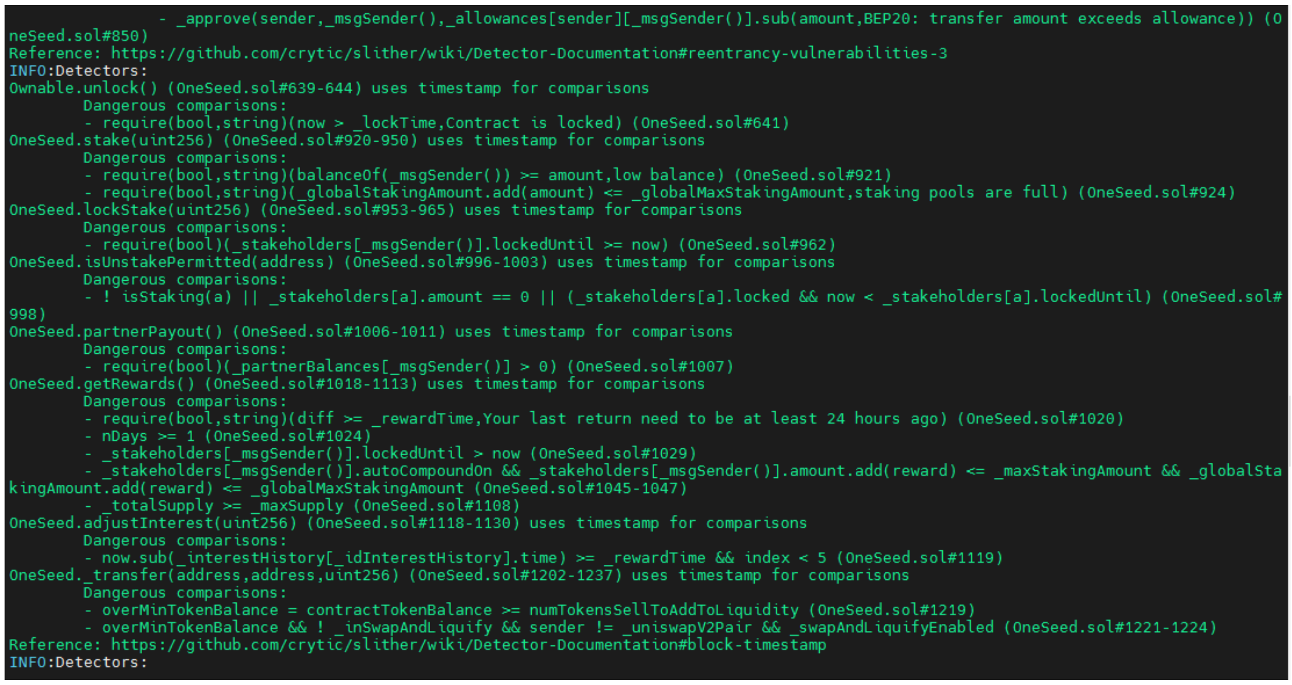
```
        - Ownable.owner() (OneSeed.sol#578-580) (function)
OneSeed._approve(address,address,uint256).owner (OneSeed.sol#1288) shadows:
        - Ownable.owner() (OneSeed.sol#578-580) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in OneSeed._transfer(address,address,uint256) (OneSeed.sol#1202-1237):
        External calls:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        State variables written after the call(s):
        - _liquidityRewardFeeTotal = _liquidityRewardFeeTotal.add(fee) (OneSeed.sol#1234)
Reentrancy in OneSeed.constructor() (OneSeed.sol#745-760):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),uniswapV2Router.WETH()) (OneSeed.sol#756
-757)
        State variables written after the call(s):
        - _uniswapV2Router = uniswapV2Router (OneSeed.sol#758)
Reentrancy in OneSeed.swapAndLiquify(uint256) (OneSeed.sol#1133-1141):
        External calls:
        - swapTokensForEth(half) (OneSeed.sol#1137)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
```



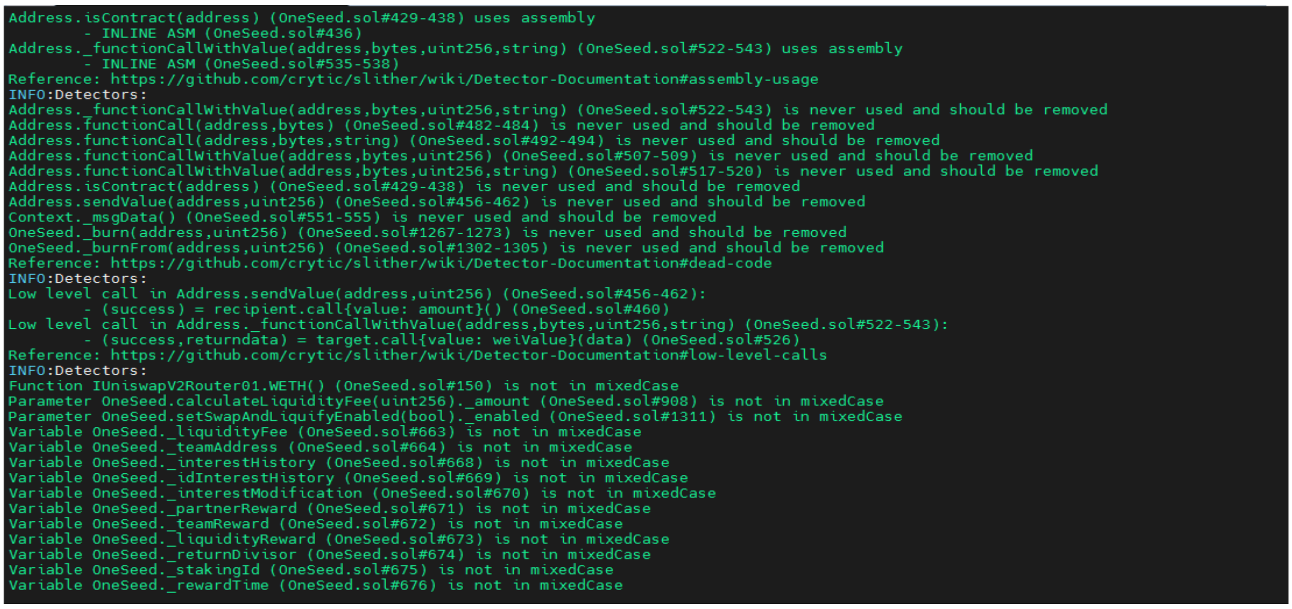
```
                - _allowances[owner][spender] = amount (OneSeed.sol#1292)
Reentrancy in OneSeed.transferFrom(address,address,uint256) (OneSeed.sol#848-852):
        External calls:
        - _transfer(sender,recipient,amount) (OneSeed.sol#849)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (OneSeed.sol#849)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (OneSeed.s
ol#850)
                - _allowances[owner][spender] = amount (OneSeed.sol#1292)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in OneSeed._transfer(address,address,uint256) (OneSeed.sol#1202-1237):
        External calls:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (OneSeed.sol#1227)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
```



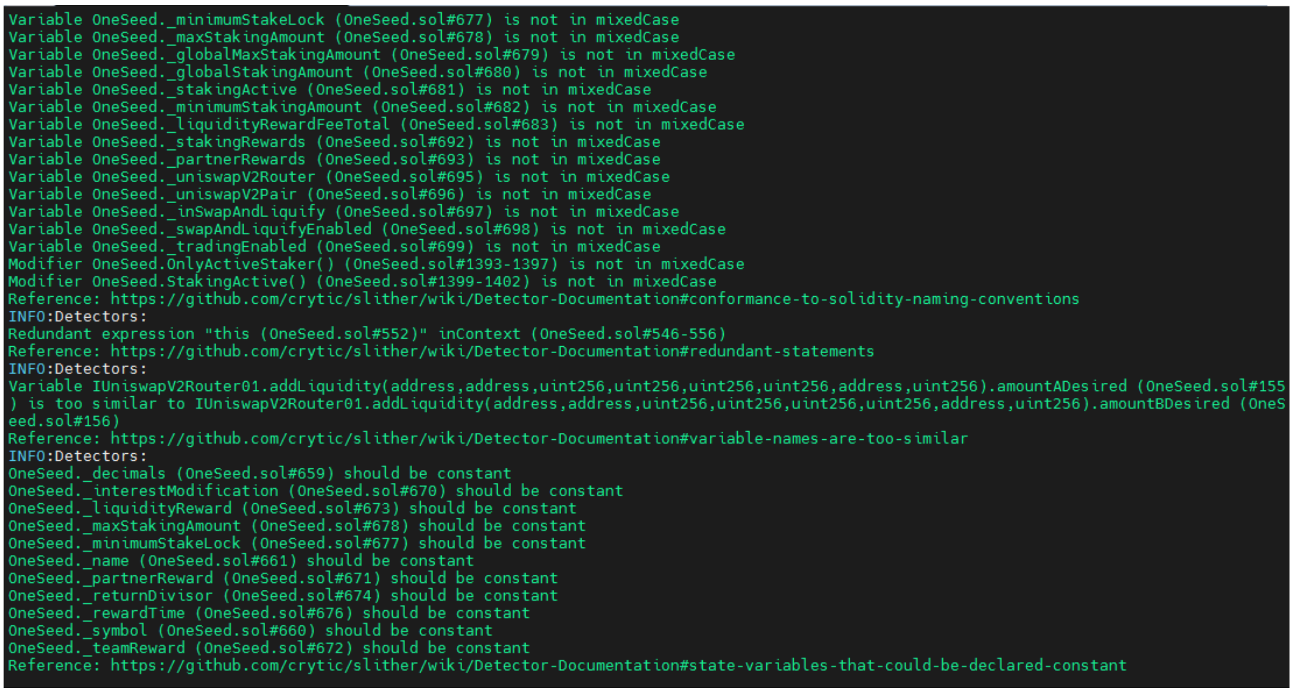
```
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (OneSeed.sol#1236)
Reentrancy in OneSeed.constructor() (OneSeed.sol#745-760):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),uniswapV2Router.WETH()) (OneSeed.sol#756
-757)
        Event emitted after the call(s):
        - Transfer(address(0x0),_msgSender(),_totalSupply) (OneSeed.sol#759)
Reentrancy in OneSeed.swapAndLiquify(uint256) (OneSeed.sol#1133-1141):
        External calls:
        - swapTokensForEth(half) (OneSeed.sol#1137)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (OneSeed.sol#1293)
                - addLiquidity(otherHalf,newBalance) (OneSeed.sol#1139)
        - SwapAndLiquify(half,newBalance,otherHalf) (OneSeed.sol#1140)
Reentrancy in OneSeed.transferFrom(address,address,uint256) (OneSeed.sol#848-852):
        External calls:
        - _transfer(sender,recipient,amount) (OneSeed.sol#849)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
                - _uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (
OneSeed.sol#1150-1156)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (OneSeed.sol#849)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (OneSeed.sol#
1161-1168)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (OneSeed.sol#1293)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (O
```



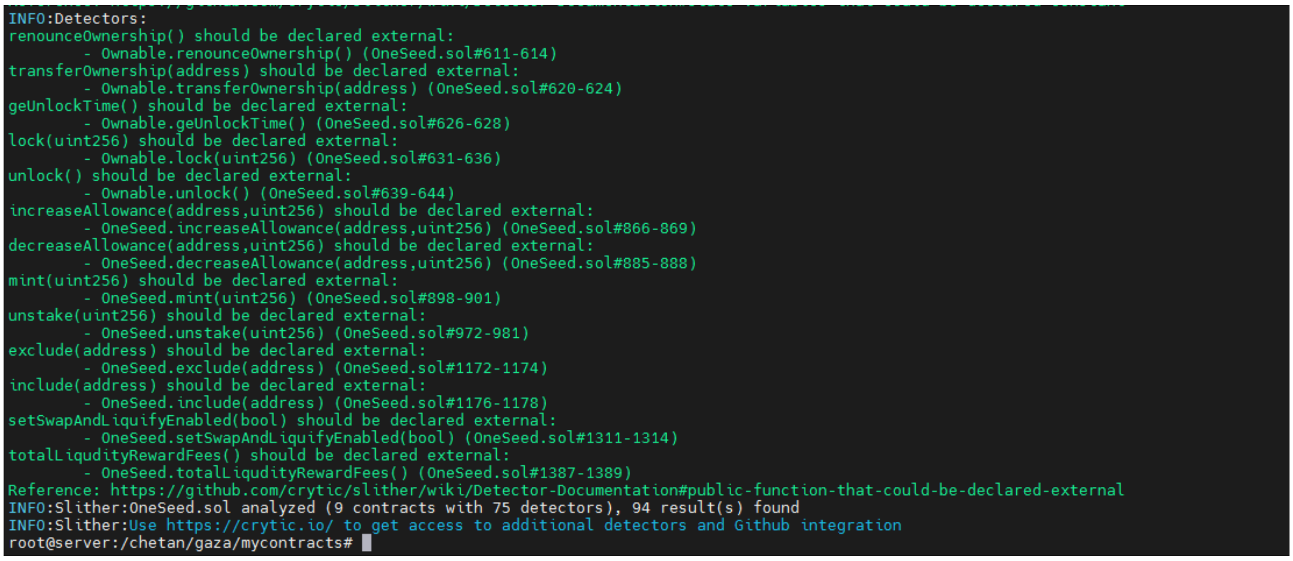
```
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (O
neSeed.sol#850)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (OneSeed.sol#639-644) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked) (OneSeed.sol#641)
OneSeed.stake(uint256) (OneSeed.sol#920-950) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(balanceOf(_msgSender()) >= amount,low balance) (OneSeed.sol#921)
        - require(bool,string)(_globalStakingAmount.add(amount) <= _globalMaxStakingAmount,staking pools are full) (OneSeed.sol#924)
OneSeed.lockStake(uint256) (OneSeed.sol#953-965) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(_stakeholders[_msgSender()].lockedUntil >= now) (OneSeed.sol#962)
OneSeed.isUnstakePermitted(address) (OneSeed.sol#996-1003) uses timestamp for comparisons
        Dangerous comparisons:
        - ! isStaking(a) || _stakeholders[a].amount == 0 || (_stakeholders[a].locked && now < _stakeholders[a].lockedUntil) (OneSeed.sol#
998)
OneSeed.partnerPayout() (OneSeed.sol#1006-1011) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(_partnerBalances[_msgSender()] > 0) (OneSeed.sol#1007)
OneSeed.getRewards() (OneSeed.sol#1018-1113) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(diff >= _rewardTime,Your last return need to be at least 24 hours ago) (OneSeed.sol#1020)
        - nDays >= 1 (OneSeed.sol#1024)
        - _stakeholders[_msgSender()].lockedUntil > now (OneSeed.sol#1029)
        - _stakeholders[_msgSender()].autoCompoundOn && _stakeholders[_msgSender()].amount.add(reward) <= _maxStakingAmount && _globalSta
kingAmount.add(reward) <= _globalMaxStakingAmount (OneSeed.sol#1045-1047)
        - _totalSupply >= _maxSupply (OneSeed.sol#1108)
OneSeed.adjustInterest(uint256) (OneSeed.sol#1118-1130) uses timestamp for comparisons
        Dangerous comparisons:
        - now.sub(_interestHistory[_idInterestHistory].time) >= _rewardTime && index < 5 (OneSeed.sol#1119)
OneSeed._transfer(address,address,uint256) (OneSeed.sol#1202-1237) uses timestamp for comparisons
        Dangerous comparisons:
        - overMinTokenBalance = contractTokenBalance >= numTokensSellToAddToLiquidity (OneSeed.sol#1219)
        - overMinTokenBalance && ! _inSwapAndLiquify && sender != _uniswapV2Pair && _swapAndLiquifyEnabled (OneSeed.sol#1221-1224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
```



```
Address.isContract(address) (OneSeed.sol#429-438) uses assembly
        - INLINE ASM (OneSeed.sol#436)
Address._functionCallWithValue(address,bytes,uint256,string) (OneSeed.sol#522-543) uses assembly
        - INLINE ASM (OneSeed.sol#535-538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (OneSeed.sol#522-543) is never used and should be removed
Address.functionCall(address,bytes) (OneSeed.sol#482-484) is never used and should be removed
Address.functionCall(address,bytes,string) (OneSeed.sol#492-494) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (OneSeed.sol#507-509) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (OneSeed.sol#517-520) is never used and should be removed
Address.isContract(address) (OneSeed.sol#429-438) is never used and should be removed
Address.sendValue(address,uint256) (OneSeed.sol#456-462) is never used and should be removed
Context._msgData() (OneSeed.sol#551-555) is never used and should be removed
OneSeed._burn(address,uint256) (OneSeed.sol#1267-1273) is never used and should be removed
OneSeed._burnFrom(address,uint256) (OneSeed.sol#1302-1305) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (OneSeed.sol#456-462):
        - (success) = recipient.call{value: amount}() (OneSeed.sol#460)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (OneSeed.sol#522-543):
        - (success,returndata) = target.call{value: weiValue}(data) (OneSeed.sol#526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Router01.WETH() (OneSeed.sol#150) is not in mixedCase
Parameter OneSeed.calculateLiquidityFee(uint256)._amount (OneSeed.sol#908) is not in mixedCase
Parameter OneSeed.setSwapAndLiquifyEnabled(bool)._enabled (OneSeed.sol#1311) is not in mixedCase
Variable OneSeed._liquidityFee (OneSeed.sol#663) is not in mixedCase
Variable OneSeed._teamAddress (OneSeed.sol#664) is not in mixedCase
Variable OneSeed._interestHistory (OneSeed.sol#668) is not in mixedCase
Variable OneSeed._idInterestHistory (OneSeed.sol#669) is not in mixedCase
Variable OneSeed._interestModification (OneSeed.sol#670) is not in mixedCase
Variable OneSeed._partnerReward (OneSeed.sol#671) is not in mixedCase
Variable OneSeed._teamReward (OneSeed.sol#672) is not in mixedCase
Variable OneSeed._liquidityReward (OneSeed.sol#673) is not in mixedCase
Variable OneSeed._returnDivisor (OneSeed.sol#674) is not in mixedCase
Variable OneSeed._stakingId (OneSeed.sol#675) is not in mixedCase
Variable OneSeed._rewardTime (OneSeed.sol#676) is not in mixedCase
```

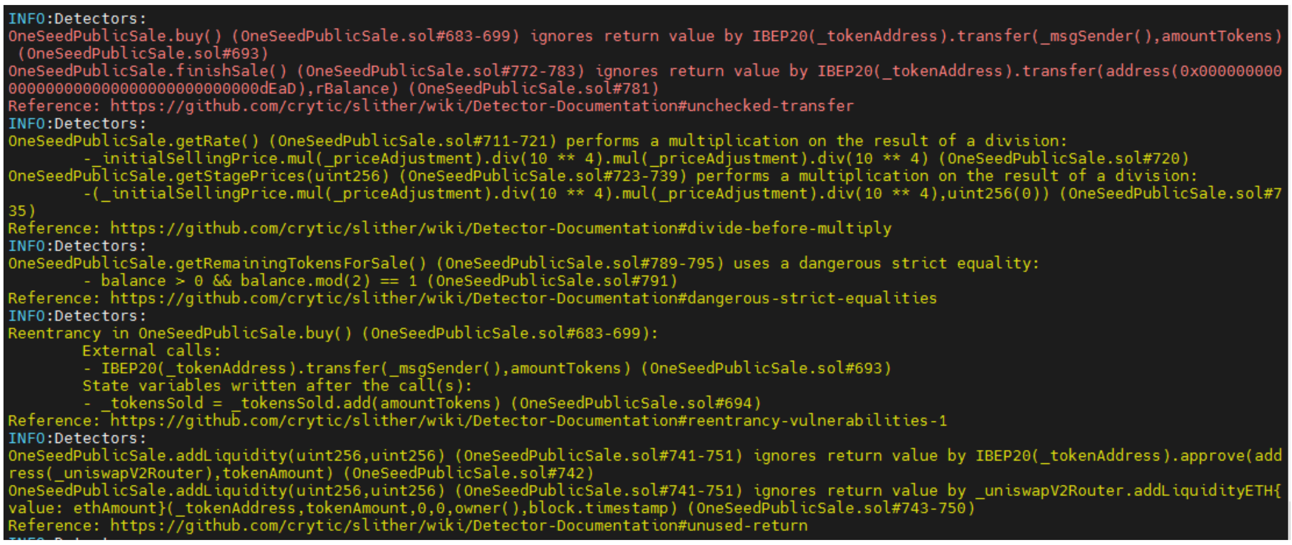


```
Variable OneSeed._minimumStakeLock (OneSeed.sol#677) is not in mixedCase
Variable OneSeed._maxStakingAmount (OneSeed.sol#678) is not in mixedCase
Variable OneSeed._globalMaxStakingAmount (OneSeed.sol#679) is not in mixedCase
Variable OneSeed._globalStakingAmount (OneSeed.sol#680) is not in mixedCase
Variable OneSeed._stakingActive (OneSeed.sol#681) is not in mixedCase
Variable OneSeed._minimumStakingAmount (OneSeed.sol#682) is not in mixedCase
Variable OneSeed._liquidityRewardFeeTotal (OneSeed.sol#683) is not in mixedCase
Variable OneSeed._stakingRewards (OneSeed.sol#692) is not in mixedCase
Variable OneSeed._partnerRewards (OneSeed.sol#693) is not in mixedCase
Variable OneSeed._uniswapV2Router (OneSeed.sol#695) is not in mixedCase
Variable OneSeed._uniswapV2Pair (OneSeed.sol#696) is not in mixedCase
Variable OneSeed._inSwapAndLiquify (OneSeed.sol#697) is not in mixedCase
Variable OneSeed._swapAndLiquifyEnabled (OneSeed.sol#698) is not in mixedCase
Variable OneSeed._tradingEnabled (OneSeed.sol#699) is not in mixedCase
Modifier OneSeed.OnlyActiveStaker() (OneSeed.sol#1393-1397) is not in mixedCase
Modifier OneSeed.StakingActive() (OneSeed.sol#1399-1402) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (OneSeed.sol#552)" inContext (OneSeed.sol#546-556)
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 (OneSeed.sol#155
) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (OneS
eed.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
OneSeed._decimals (OneSeed.sol#659) should be constant
OneSeed._interestModification (OneSeed.sol#670) should be constant
OneSeed._liquidityReward (OneSeed.sol#673) should be constant
OneSeed._maxStakingAmount (OneSeed.sol#678) should be constant
OneSeed._minimumStakeLock (OneSeed.sol#677) should be constant
OneSeed._name (OneSeed.sol#661) should be constant
OneSeed._partnerReward (OneSeed.sol#671) should be constant
OneSeed._returnDivisor (OneSeed.sol#674) should be constant
OneSeed._rewardTime (OneSeed.sol#676) should be constant
OneSeed._symbol (OneSeed.sol#660) should be constant
OneSeed._teamReward (OneSeed.sol#672) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

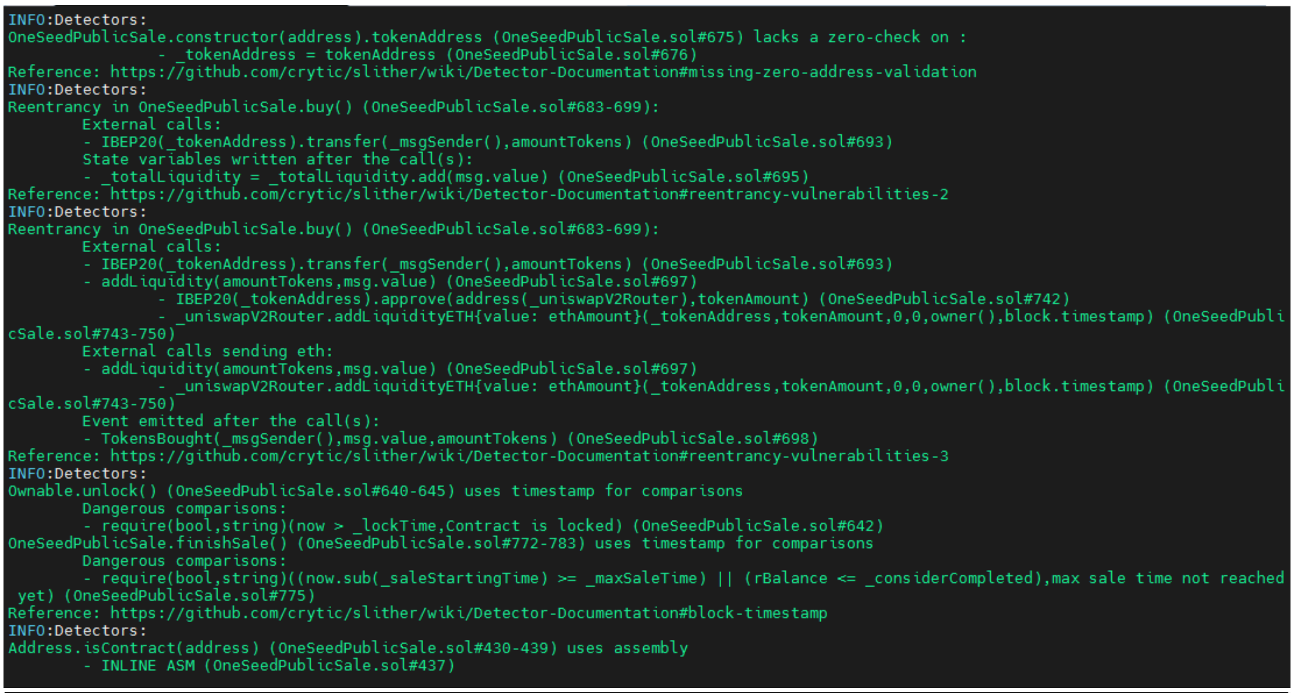


```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (OneSeed.sol#611-614)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (OneSeed.sol#620-624)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (OneSeed.sol#626-628)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (OneSeed.sol#631-636)
unlock() should be declared external:
        - Ownable.unlock() (OneSeed.sol#639-644)
increaseAllowance(address,uint256) should be declared external:
        - OneSeed.increaseAllowance(address,uint256) (OneSeed.sol#866-869)
decreaseAllowance(address,uint256) should be declared external:
        - OneSeed.decreaseAllowance(address,uint256) (OneSeed.sol#885-888)
mint(uint256) should be declared external:
        - OneSeed.mint(uint256) (OneSeed.sol#898-901)
unstake(uint256) should be declared external:
        - OneSeed.unstake(uint256) (OneSeed.sol#972-981)
exclude(address) should be declared external:
        - OneSeed.exclude(address) (OneSeed.sol#1172-1174)
include(address) should be declared external:
        - OneSeed.include(address) (OneSeed.sol#1176-1178)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - OneSeed.setSwapAndLiquifyEnabled(bool) (OneSeed.sol#1311-1314)
totalLiqudityRewardFees() should be declared external:
        - OneSeed.totalLiqudityRewardFees() (OneSeed.sol#1387-1389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:OneSeed.sol analyzed (9 contracts with 75 detectors), 94 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

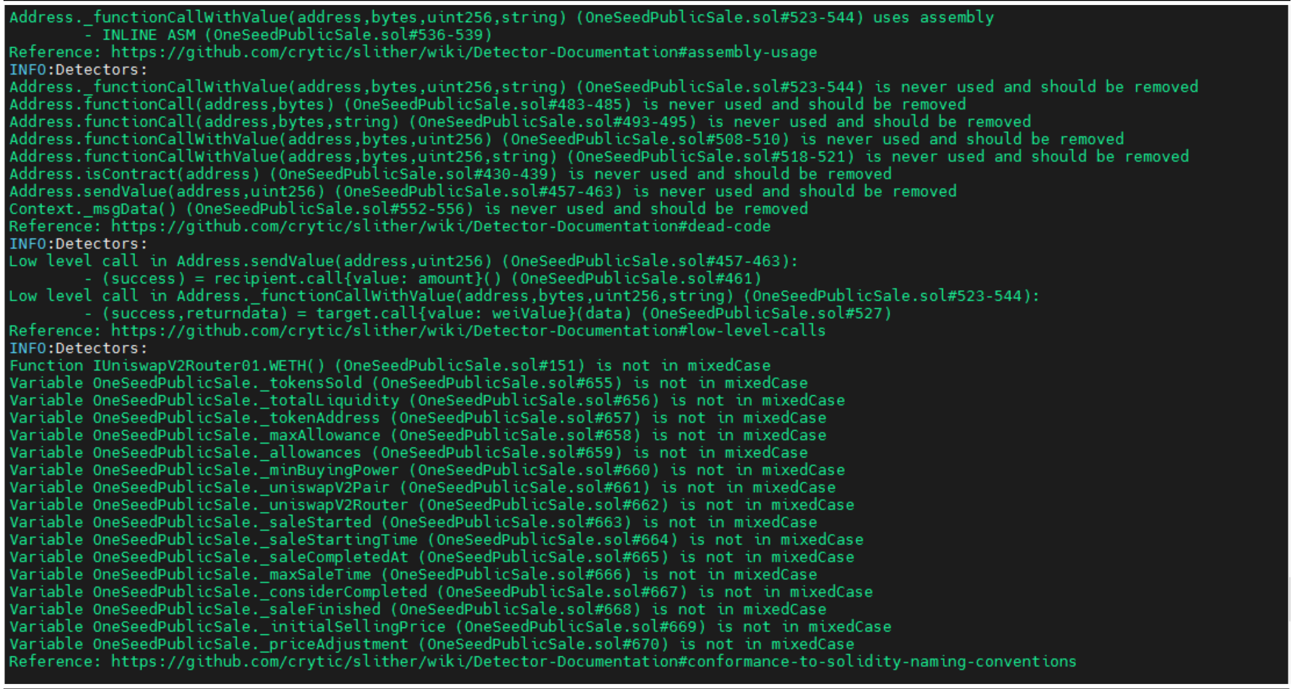
Slither log >> OneSeedPublicSale.sol



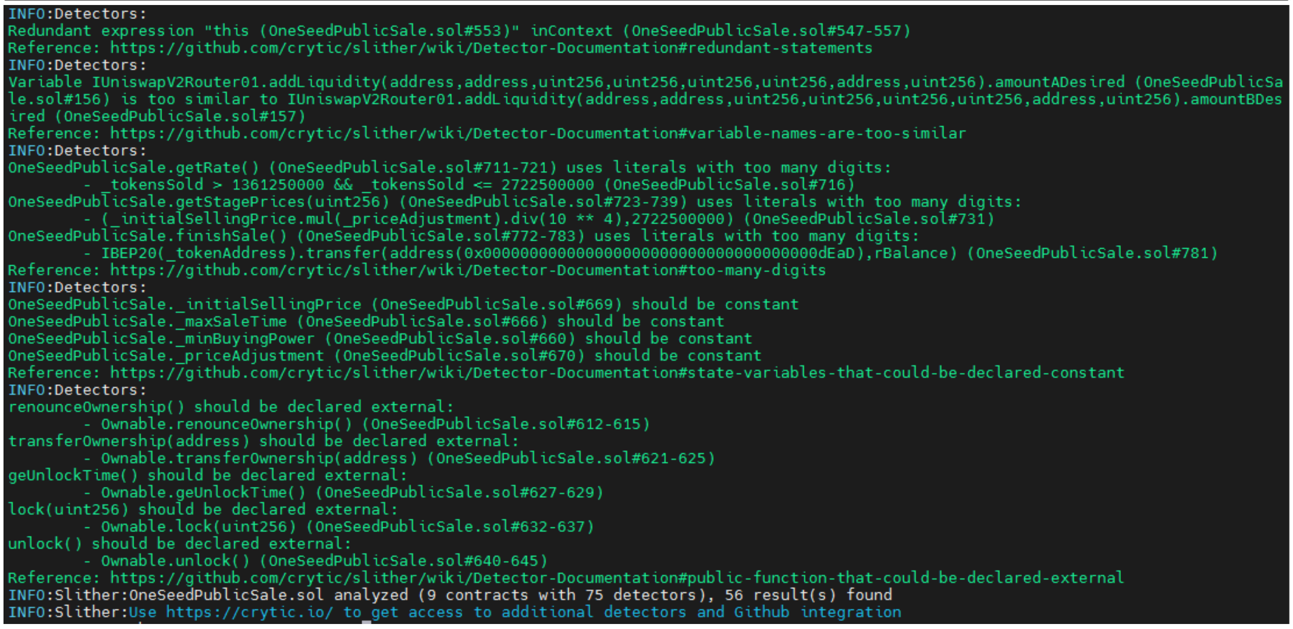
```
INFO:Detectors:
OneSeedPublicSale.buy() (OneSeedPublicSale.sol#683-699) ignores return value by IBEP20(_tokenAddress).transfer(_msgSender(),amountTokens)
 (OneSeedPublicSale.sol#693)
OneSeedPublicSale.finishSale() (OneSeedPublicSale.sol#772-783) ignores return value by IBEP20(_tokenAddress).transfer(address(0x000000000
000000000000000000000000000dEaD),rBalance) (OneSeedPublicSale.sol#781)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
OneSeedPublicSale.getRate() (OneSeedPublicSale.sol#711-721) performs a multiplication on the result of a division:
        -_initialSellingPrice.mul(_priceAdjustment).div(10 ** 4).mul(_priceAdjustment).div(10 ** 4) (OneSeedPublicSale.sol#720)
OneSeedPublicSale.getStagePrices(uint256) (OneSeedPublicSale.sol#723-739) performs a multiplication on the result of a division:
        -(_initialSellingPrice.mul(_priceAdjustment).div(10 ** 4).mul(_priceAdjustment).div(10 ** 4),uint256(0)) (OneSeedPublicSale.sol#7
35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
OneSeedPublicSale.getRemainingTokensForSale() (OneSeedPublicSale.sol#789-795) uses a dangerous strict equality:
        - balance > 0 && balance.mod(2) == 1 (OneSeedPublicSale.sol#791)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in OneSeedPublicSale.buy() (OneSeedPublicSale.sol#683-699):
        External calls:
        - IBEP20(_tokenAddress).transfer(_msgSender(),amountTokens) (OneSeedPublicSale.sol#693)
        State variables written after the call(s):
        - _tokensSold = _tokensSold.add(amountTokens) (OneSeedPublicSale.sol#694)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
OneSeedPublicSale.addLiquidity(uint256,uint256) (OneSeedPublicSale.sol#741-751) ignores return value by IBEP20(_tokenAddress).approve(add
ress(_uniswapV2Router),tokenAmount) (OneSeedPublicSale.sol#742)
OneSeedPublicSale.addLiquidity(uint256,uint256) (OneSeedPublicSale.sol#741-751) ignores return value by _uniswapV2Router.addLiquidityETH{
value: ethAmount}(_tokenAddress,tokenAmount,0,0,owner(),block.timestamp) (OneSeedPublicSale.sol#743-750)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```



```
INFO:Detectors:
OneSeedPublicSale.constructor(address).tokenAddress (OneSeedPublicSale.sol#675) lacks a zero-check on :
                - _tokenAddress = tokenAddress (OneSeedPublicSale.sol#676)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in OneSeedPublicSale.buy() (OneSeedPublicSale.sol#683-699):
        External calls:
        - IBEP20(_tokenAddress).transfer(_msgSender(),amountTokens) (OneSeedPublicSale.sol#693)
        State variables written after the call(s):
        - _totalLiquidity = _totalLiquidity.add(msg.value) (OneSeedPublicSale.sol#695)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in OneSeedPublicSale.buy() (OneSeedPublicSale.sol#683-699):
        External calls:
        - IBEP20(_tokenAddress).transfer(_msgSender(),amountTokens) (OneSeedPublicSale.sol#693)
        - addLiquidity(amountTokens,msg.value) (OneSeedPublicSale.sol#697)
                - IBEP20(_tokenAddress).approve(address(_uniswapV2Router),tokenAmount) (OneSeedPublicSale.sol#742)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(_tokenAddress,tokenAmount,0,0,owner(),block.timestamp) (OneSeedPubli
cSale.sol#743-750)
        External calls sending eth:
        - addLiquidity(amountTokens,msg.value) (OneSeedPublicSale.sol#697)
                - _uniswapV2Router.addLiquidityETH{value: ethAmount}(_tokenAddress,tokenAmount,0,0,owner(),block.timestamp) (OneSeedPubli
cSale.sol#743-750)
        Event emitted after the call(s):
        - TokensBought(_msgSender(),msg.value,amountTokens) (OneSeedPublicSale.sol#698)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (OneSeedPublicSale.sol#640-645) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked) (OneSeedPublicSale.sol#642)
OneSeedPublicSale.finishSale() (OneSeedPublicSale.sol#772-783) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((now.sub(_saleStartingTime) >= _maxSaleTime) || (rBalance <= _considerCompleted),max sale time not reached
 yet) (OneSeedPublicSale.sol#775)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (OneSeedPublicSale.sol#430-439) uses assembly
        - INLINE ASM (OneSeedPublicSale.sol#437)
```



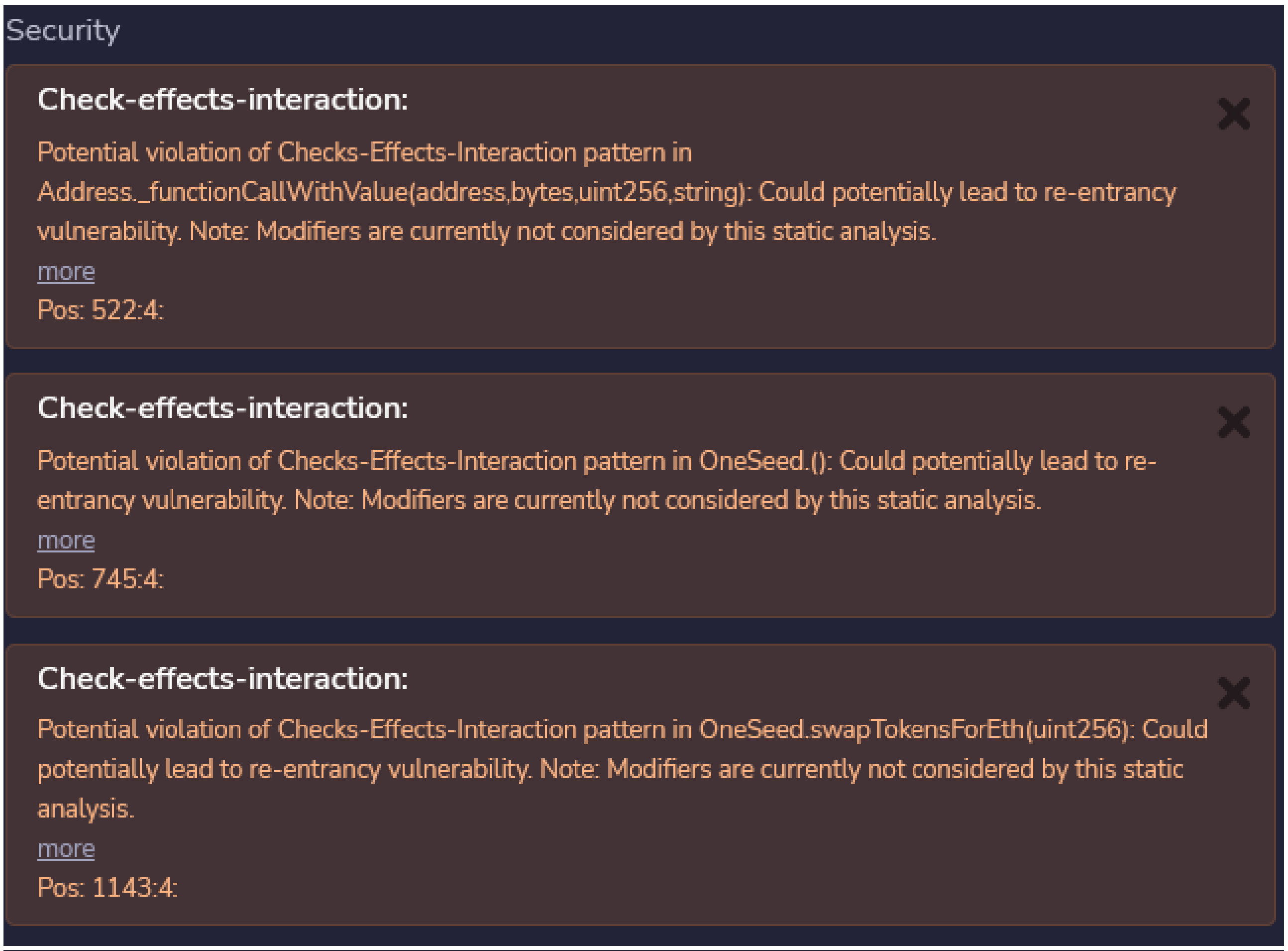
```
Address._functionCallWithValue(address,bytes,uint256,string) (OneSeedPublicSale.sol#523-544) uses assembly
        - INLINE ASM (OneSeedPublicSale.sol#536-539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (OneSeedPublicSale.sol#523-544) is never used and should be removed
Address.functionCall(address,bytes) (OneSeedPublicSale.sol#483-485) is never used and should be removed
Address.functionCall(address,bytes,string) (OneSeedPublicSale.sol#493-495) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (OneSeedPublicSale.sol#508-510) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (OneSeedPublicSale.sol#518-521) is never used and should be removed
Address.isContract(address) (OneSeedPublicSale.sol#430-439) is never used and should be removed
Address.sendValue(address,uint256) (OneSeedPublicSale.sol#457-463) is never used and should be removed
Context._msgData() (OneSeedPublicSale.sol#552-556) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (OneSeedPublicSale.sol#457-463):
        - (success) = recipient.call{value: amount}() (OneSeedPublicSale.sol#461)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (OneSeedPublicSale.sol#523-544):
        - (success,returndata) = target.call{value: weiValue}(data) (OneSeedPublicSale.sol#527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Router01.WETH() (OneSeedPublicSale.sol#151) is not in mixedCase
Variable OneSeedPublicSale._tokensSold (OneSeedPublicSale.sol#655) is not in mixedCase
Variable OneSeedPublicSale._totalLiquidity (OneSeedPublicSale.sol#656) is not in mixedCase
Variable OneSeedPublicSale._tokenAddress (OneSeedPublicSale.sol#657) is not in mixedCase
Variable OneSeedPublicSale._maxAllowance (OneSeedPublicSale.sol#658) is not in mixedCase
Variable OneSeedPublicSale._allowances (OneSeedPublicSale.sol#659) is not in mixedCase
Variable OneSeedPublicSale._minBuyingPower (OneSeedPublicSale.sol#660) is not in mixedCase
Variable OneSeedPublicSale._uniswapV2Pair (OneSeedPublicSale.sol#661) is not in mixedCase
Variable OneSeedPublicSale._uniswapV2Router (OneSeedPublicSale.sol#662) is not in mixedCase
Variable OneSeedPublicSale._saleStarted (OneSeedPublicSale.sol#663) is not in mixedCase
Variable OneSeedPublicSale._saleStartingTime (OneSeedPublicSale.sol#664) is not in mixedCase
Variable OneSeedPublicSale._saleCompletedAt (OneSeedPublicSale.sol#665) is not in mixedCase
Variable OneSeedPublicSale._maxSaleTime (OneSeedPublicSale.sol#666) is not in mixedCase
Variable OneSeedPublicSale._considerCompleted (OneSeedPublicSale.sol#667) is not in mixedCase
Variable OneSeedPublicSale._saleFinished (OneSeedPublicSale.sol#668) is not in mixedCase
Variable OneSeedPublicSale._initialSellingPrice (OneSeedPublicSale.sol#669) is not in mixedCase
Variable OneSeedPublicSale._priceAdjustment (OneSeedPublicSale.sol#670) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
INFO:Detectors:
Redundant expression "this (OneSeedPublicSale.sol#553)" inContext (OneSeedPublicSale.sol#547-557)
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 (OneSeedPublicSa
le.sol#156) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDes
ired (OneSeedPublicSale.sol#157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
OneSeedPublicSale.getRate() (OneSeedPublicSale.sol#711-721) uses literals with too many digits:
        - _tokensSold > 1361250000 && _tokensSold <= 2722500000 (OneSeedPublicSale.sol#716)
OneSeedPublicSale.getStagePrices(uint256) (OneSeedPublicSale.sol#723-739) uses literals with too many digits:
        - (_initialSellingPrice.mul(_priceAdjustment).div(10 ** 4),2722500000) (OneSeedPublicSale.sol#731)
OneSeedPublicSale.finishSale() (OneSeedPublicSale.sol#772-783) uses literals with too many digits:
        - IBEP20(_tokenAddress).transfer(address(0x000000000000000000000000000000000000dEaD),rBalance) (OneSeedPublicSale.sol#781)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
OneSeedPublicSale._initialSellingPrice (OneSeedPublicSale.sol#669) should be constant
OneSeedPublicSale._maxSaleTime (OneSeedPublicSale.sol#666) should be constant
OneSeedPublicSale._minBuyingPower (OneSeedPublicSale.sol#660) should be constant
OneSeedPublicSale._priceAdjustment (OneSeedPublicSale.sol#670) 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() (OneSeedPublicSale.sol#612-615)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (OneSeedPublicSale.sol#621-625)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (OneSeedPublicSale.sol#627-629)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (OneSeedPublicSale.sol#632-637)
unlock() should be declared external:
        - Ownable.unlock() (OneSeedPublicSale.sol#640-645)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:OneSeedPublicSale.sol analyzed (9 contracts with 75 detectors), 56 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

OneSeed.sol



Security

Check-effects-interaction:

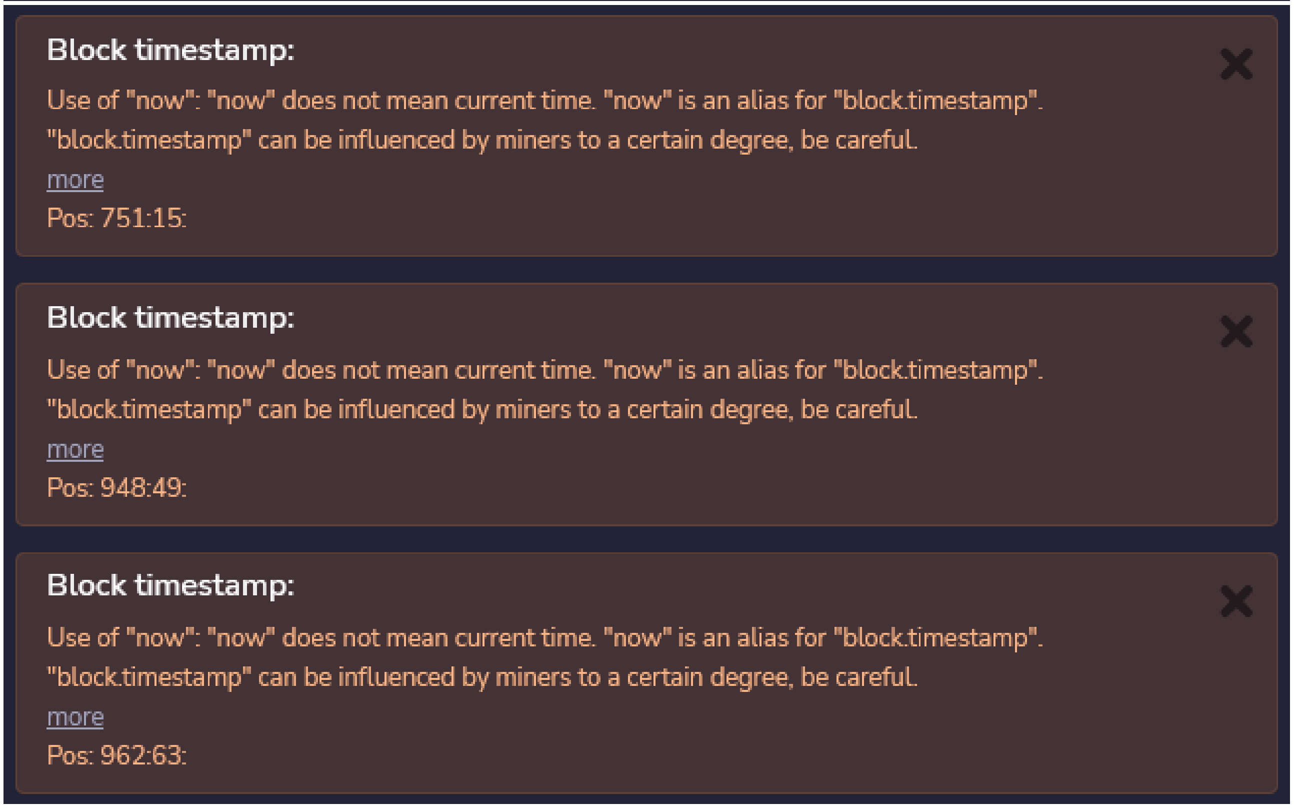
Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 522:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in OneSeed.(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 745:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in OneSeed.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1143:4:



Block timestamp:

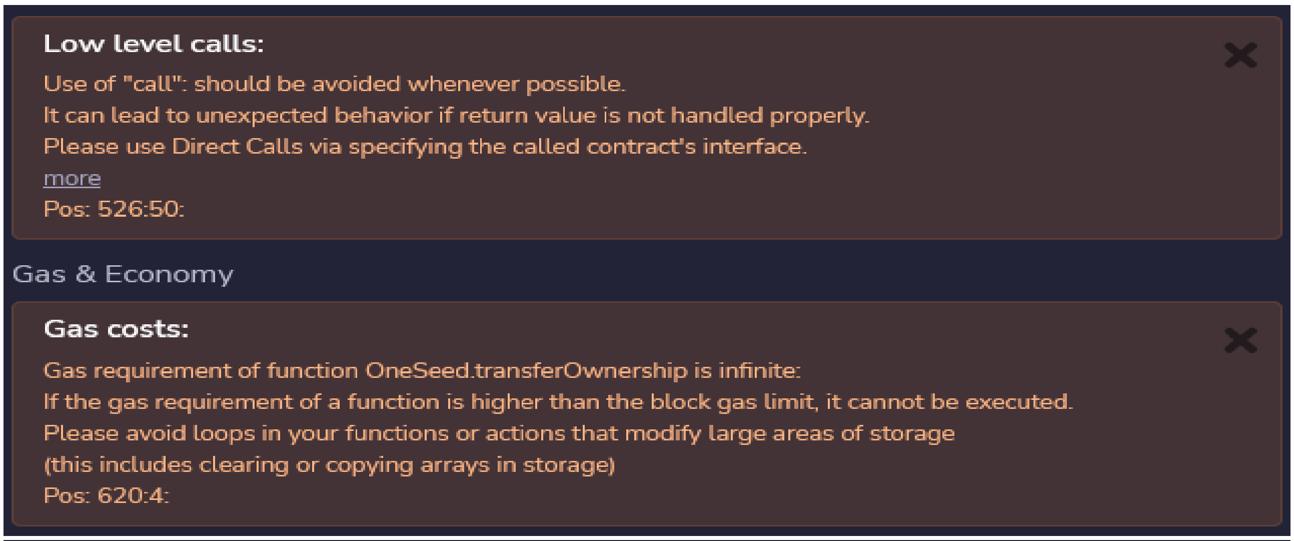
Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp". "block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 751:15:

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: 948:49:

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: 962:63:



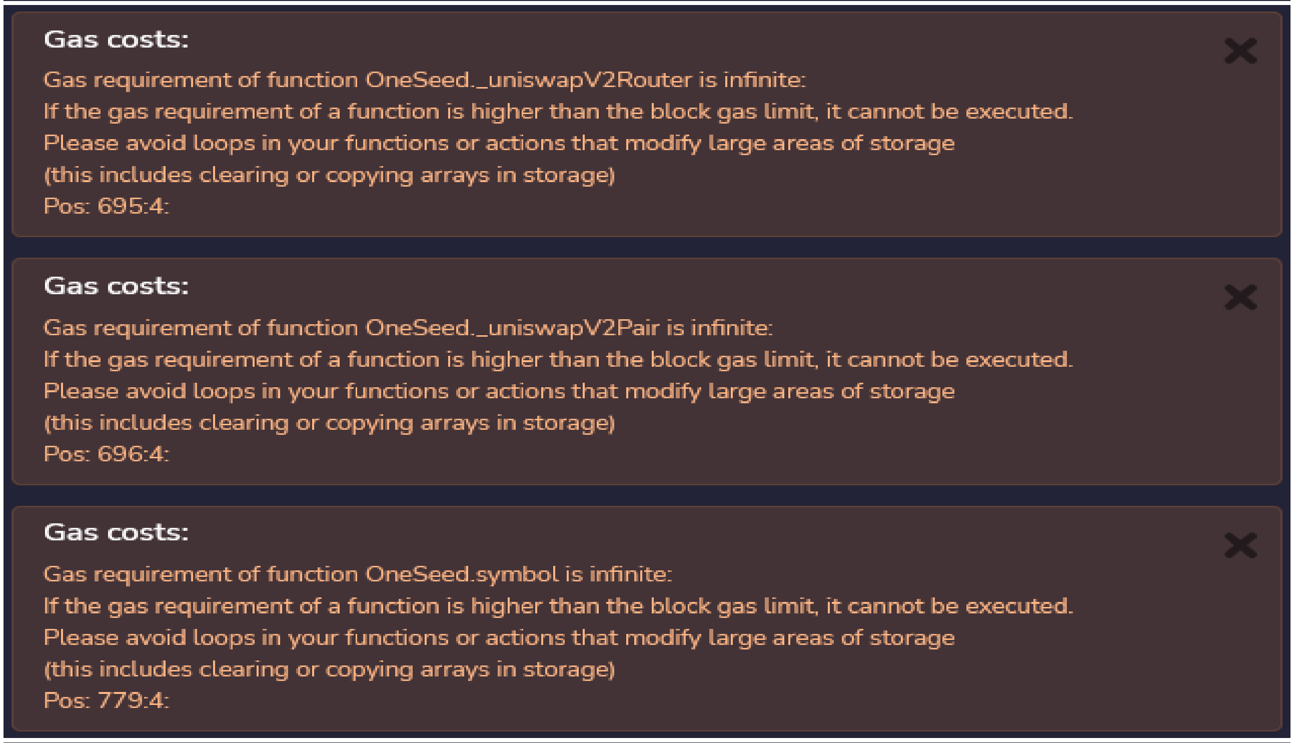
Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 526:50:

Gas & Economy

Gas costs:

Gas requirement of function OneSeed.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: 620:4:



Gas costs:

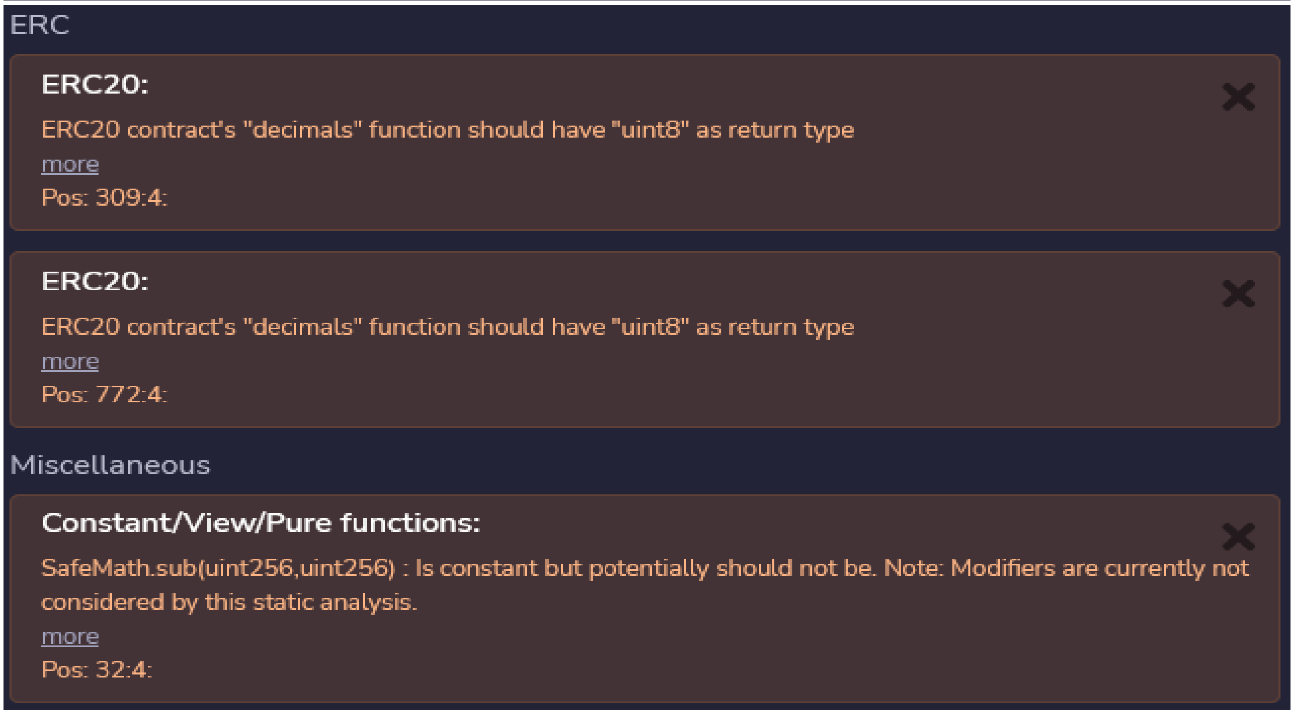
Gas requirement of function OneSeed._uniswapV2Router is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 695:4:

Gas costs:

Gas requirement of function OneSeed._uniswapV2Pair is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 696:4:

Gas costs:

Gas requirement of function OneSeed.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 779:4:



ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 309:4:

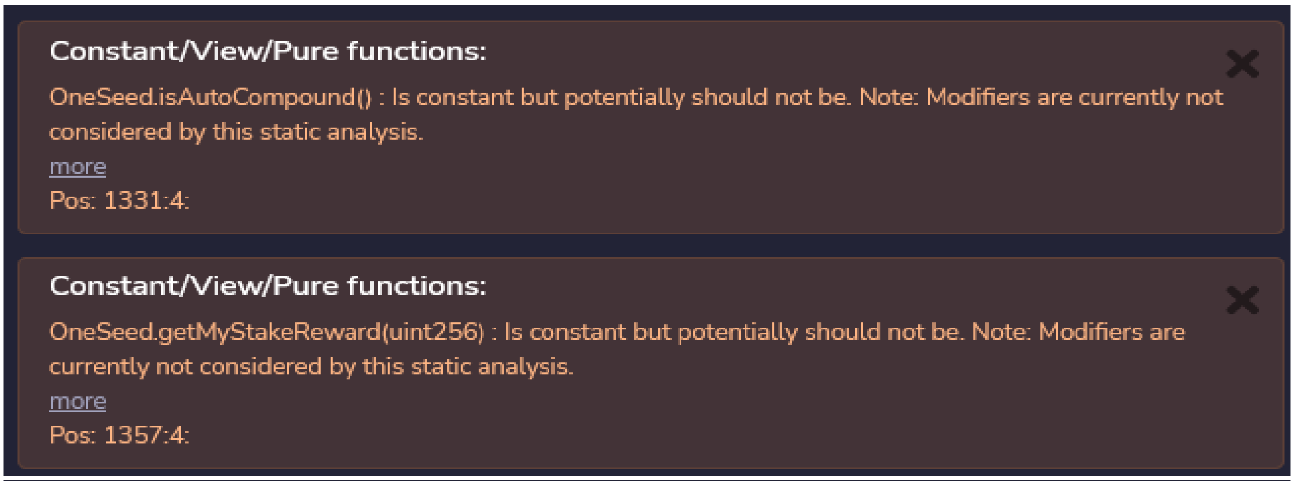
ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 772:4:

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: 32:4:



Constant/View/Pure functions:

OneSeed.isAutoCompound() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

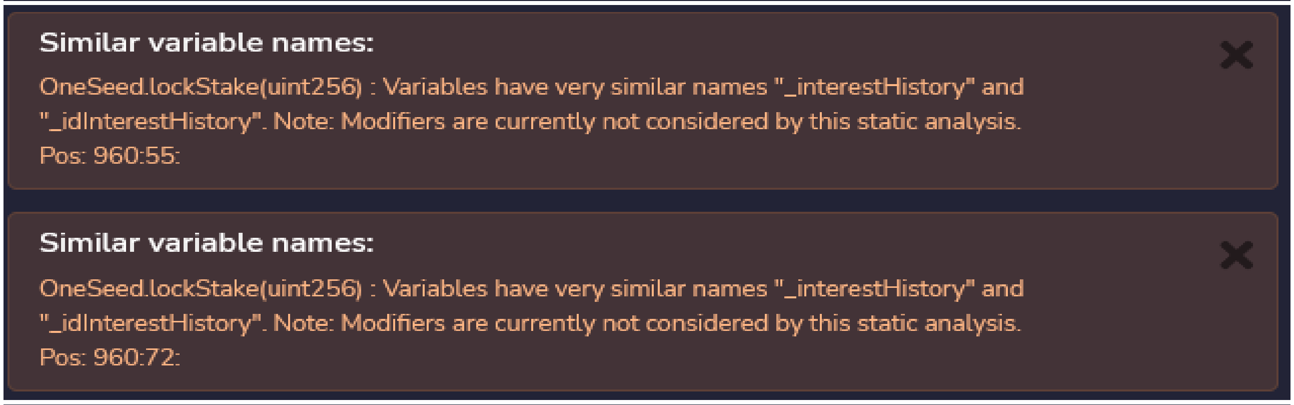
Pos: 1331:4:

Constant/View/Pure functions:

OneSeed.getMyStakeReward(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1357:4:



Similar variable names:

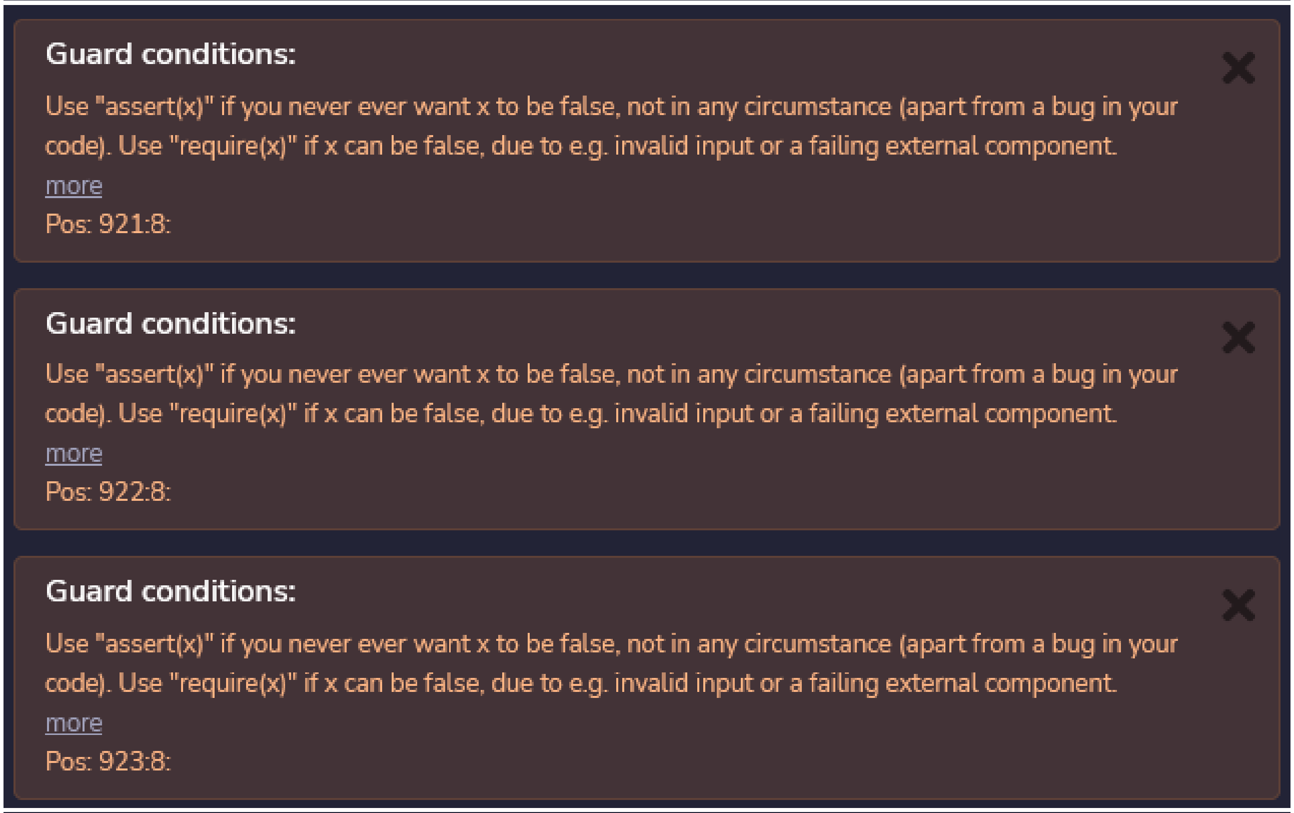
OneSeed.lockStake(uint256) : Variables have very similar names "_interestHistory" and "_idInterestHistory". Note: Modifiers are currently not considered by this static analysis.

Pos: 960:55:

Similar variable names:

OneSeed.lockStake(uint256) : Variables have very similar names "_interestHistory" and "_idInterestHistory". Note: Modifiers are currently not considered by this static analysis.

Pos: 960:72:



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: 921:8:

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

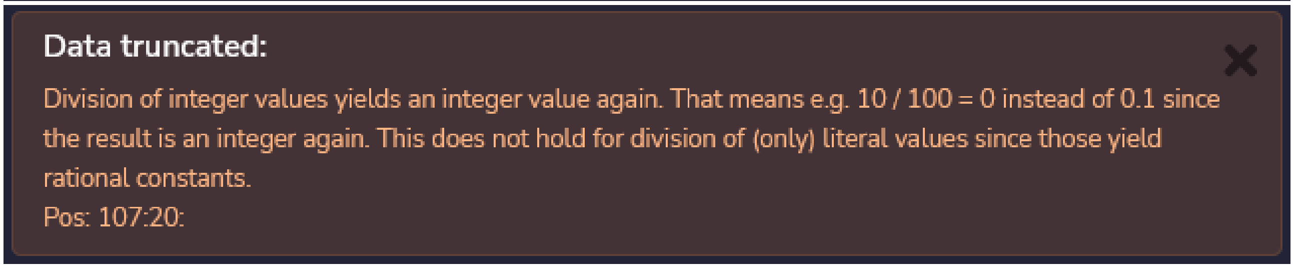
Pos: 922:8:

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: 923:8:

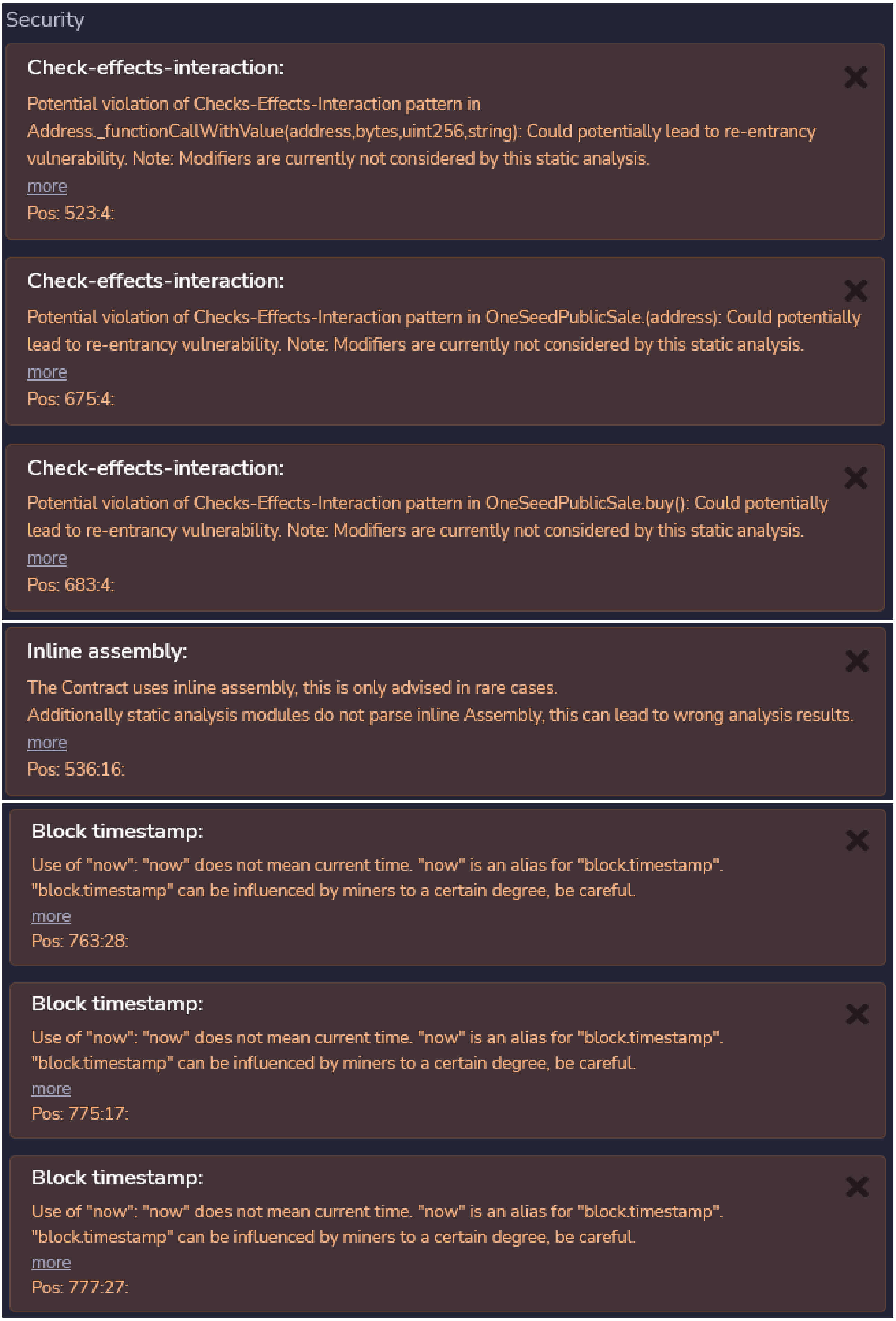


Data truncated:

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: 107:20:

OneSeedPublicSale.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 523:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in OneSeedPublicSale.(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 675:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in OneSeedPublicSale.buy(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 683:4:

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: 536:16:

Block timestamp:

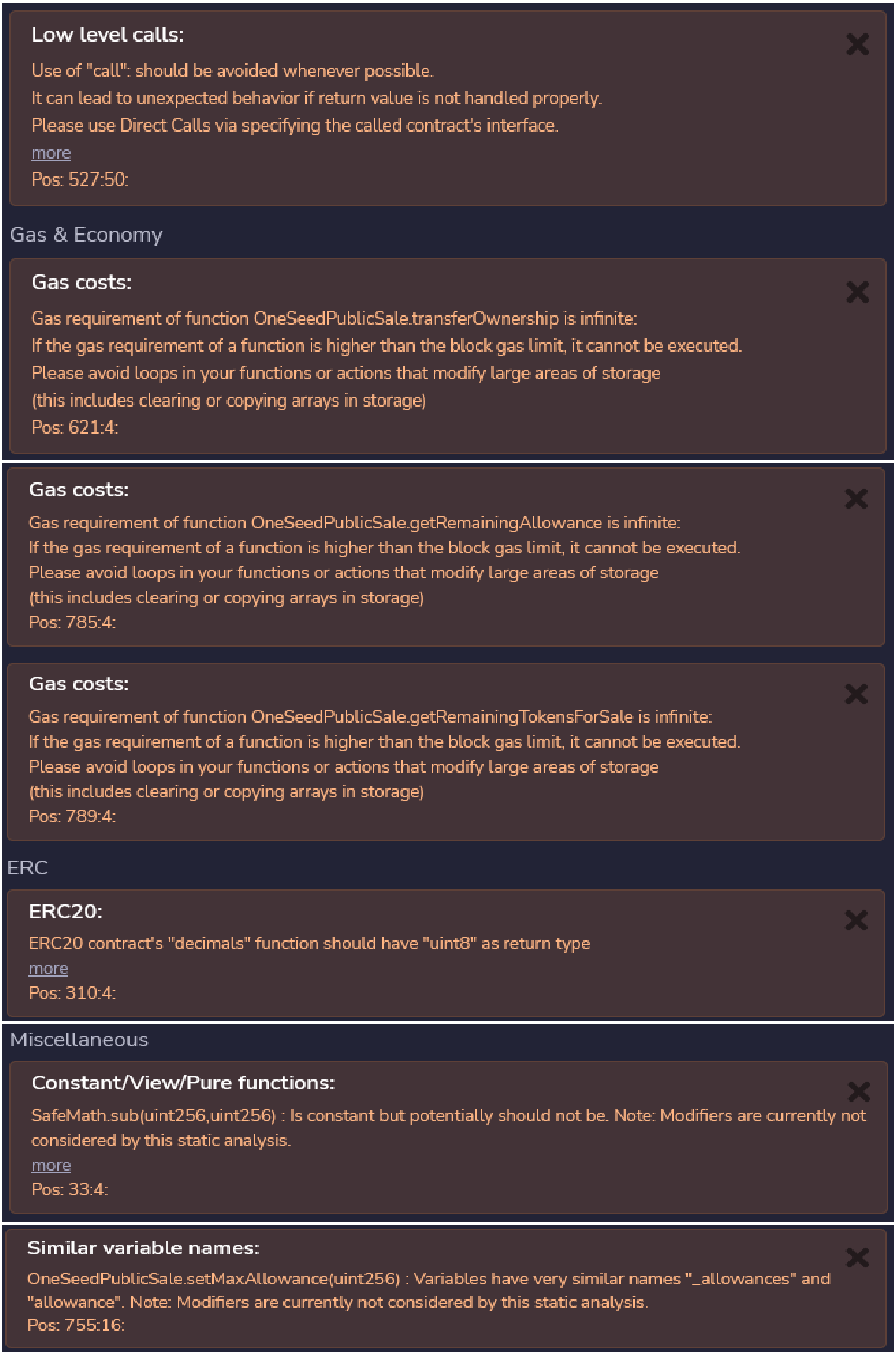
Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 763:28:

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: 775:17:

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: 777:27:



Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 527:50:

Gas & Economy

Gas costs:

Gas requirement of function OneSeedPublicSale.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: 621:4:

Gas costs:

Gas requirement of function OneSeedPublicSale.getRemainingAllowance 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: 785:4:

Gas costs:

Gas requirement of function OneSeedPublicSale.getRemainingTokensForSale 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: 789:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 310:4:

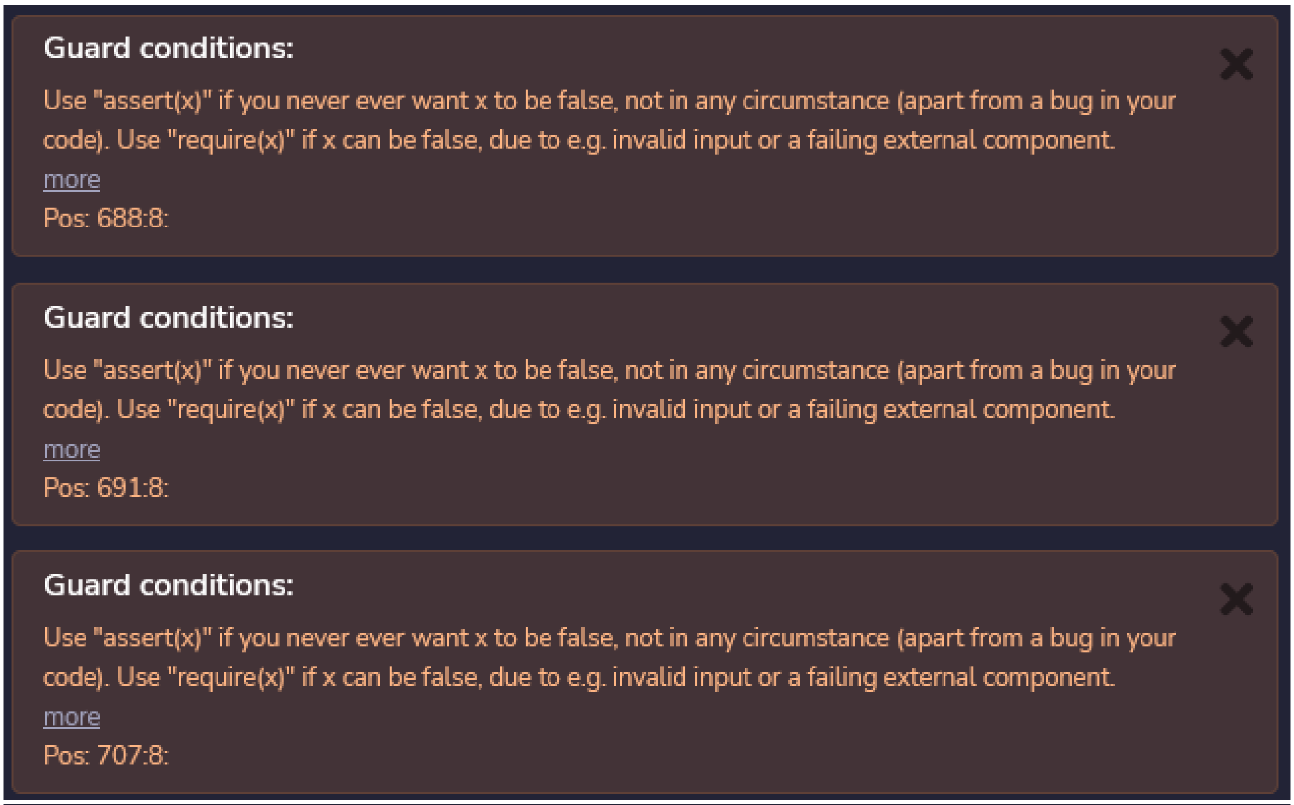
Miscellaneous

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: 33:4:

Similar variable names:

OneSeedPublicSale.setMaxAllowance(uint256) : Variables have very similar names "_allowances" and "allowance". Note: Modifiers are currently not considered by this static analysis.
Pos: 755:16:



Guard conditions:

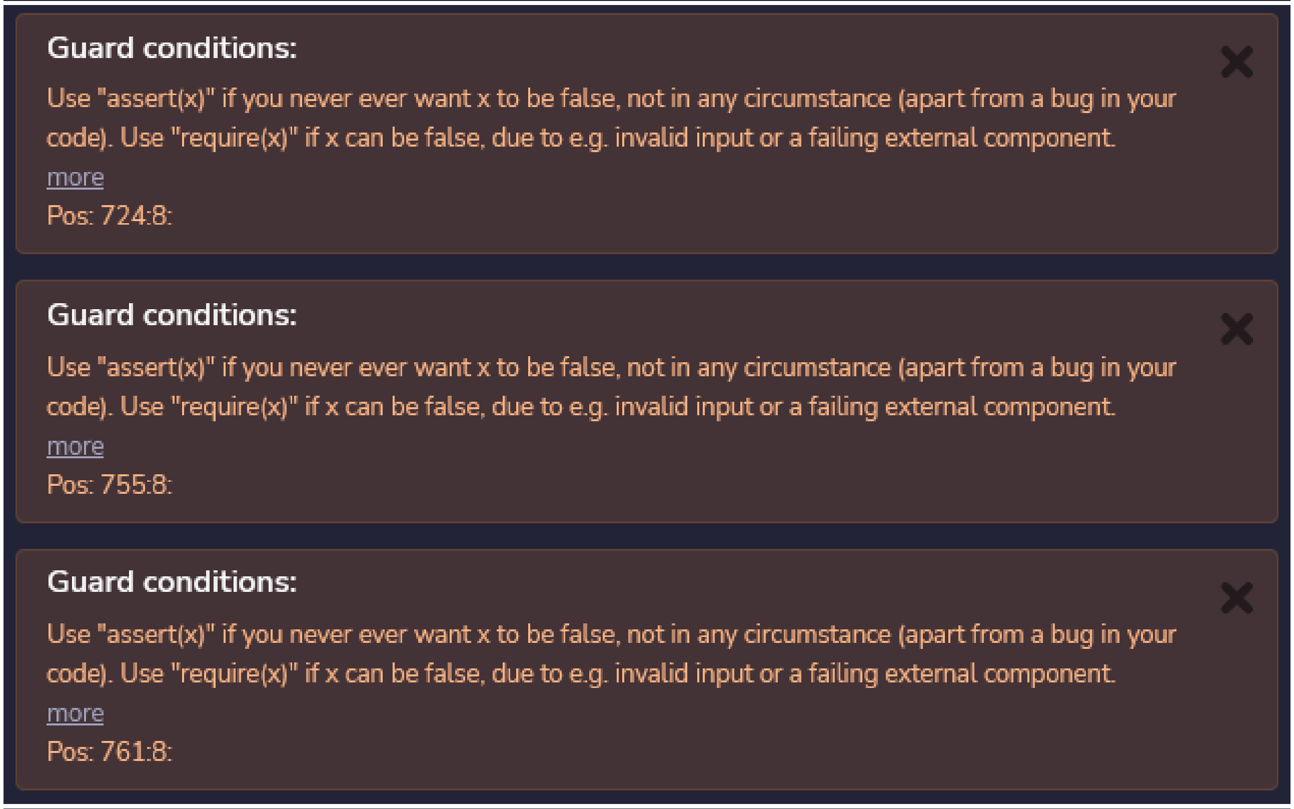
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 688:8:

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: 691:8:

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: 707:8:



Guard conditions:

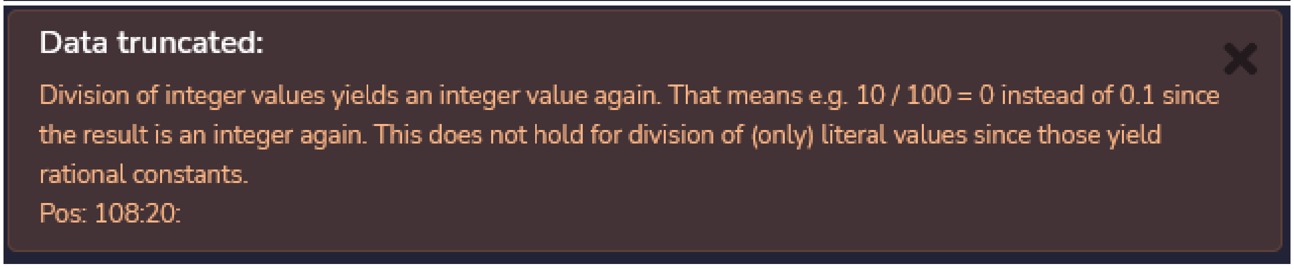
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 724:8:

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: 755:8:

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: 761:8:

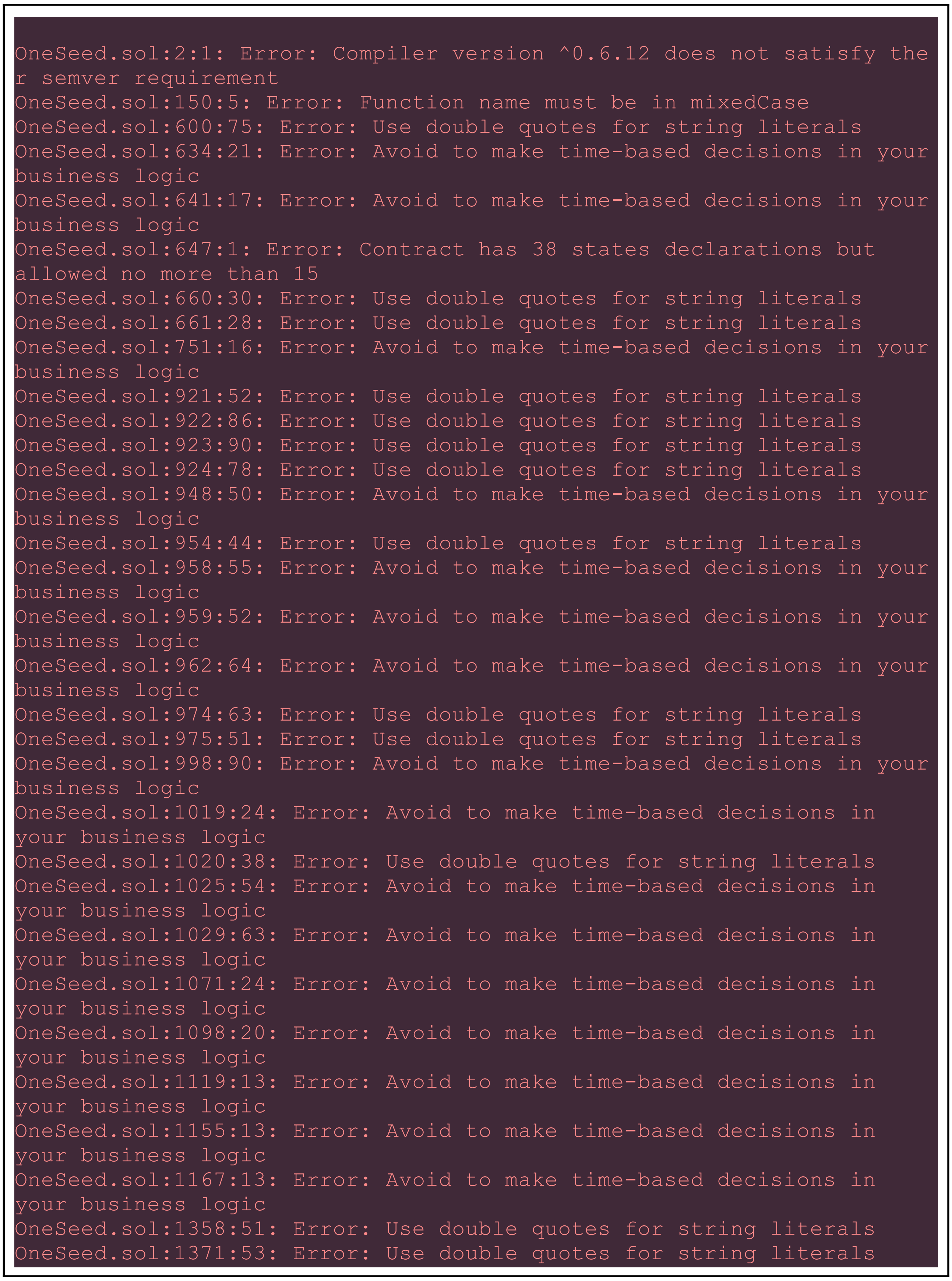


Data truncated:

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: 108:20:

Solhint Linter

OneSeed.sol



```
OneSeed.sol:2:1: Error: Compiler version ^0.6.12 does not satisfy the
r semver requirement
OneSeed.sol:150:5: Error: Function name must be in mixedCase
OneSeed.sol:600:75: Error: Use double quotes for string literals
OneSeed.sol:634:21: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:641:17: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:647:1: Error: Contract has 38 states declarations but
allowed no more than 15
OneSeed.sol:660:30: Error: Use double quotes for string literals
OneSeed.sol:661:28: Error: Use double quotes for string literals
OneSeed.sol:751:16: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:921:52: Error: Use double quotes for string literals
OneSeed.sol:922:86: Error: Use double quotes for string literals
OneSeed.sol:923:90: Error: Use double quotes for string literals
OneSeed.sol:924:78: Error: Use double quotes for string literals
OneSeed.sol:948:50: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:954:44: Error: Use double quotes for string literals
OneSeed.sol:958:55: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:959:52: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:962:64: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:974:63: Error: Use double quotes for string literals
OneSeed.sol:975:51: Error: Use double quotes for string literals
OneSeed.sol:998:90: Error: Avoid to make time-based decisions in your
business logic
OneSeed.sol:1019:24: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1020:38: Error: Use double quotes for string literals
OneSeed.sol:1025:54: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1029:63: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1071:24: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1098:20: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1119:13: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1155:13: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1167:13: Error: Avoid to make time-based decisions in
your business logic
OneSeed.sol:1358:51: Error: Use double quotes for string literals
OneSeed.sol:1371:53: Error: Use double quotes for string literals
OneSeed.sol:1391:32: Error: Code contains empty blocks
OneSeed.sol:1394:42: Error: Use double quotes for string literals
OneSeed.sol:1395:57: Error: Use double quotes for string literals
OneSeed.sol:1400:33: Error: Use double quotes for string literals
```

OneSeedPublicSale.sol

```
OneSeedPublicSale.sol:2:1: Error: Compiler version ^0.6.12 does not
satisfy the r semver requirement
OneSeedPublicSale.sol:151:5: Error: Function name must be in
mixedCase
OneSeedPublicSale.sol:601:75: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:635:21: Error: Avoid to make time-based
decisions in your business logic
OneSeedPublicSale.sol:642:17: Error: Avoid to make time-based
decisions in your business logic
OneSeedPublicSale.sol:659:5: Error: Explicitly mark visibility of
state
OneSeedPublicSale.sol:684:31: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:685:33: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:686:47: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:688:61: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:691:62: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:694:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
OneSeedPublicSale.sol:695:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
OneSeedPublicSale.sol:749:13: Error: Avoid to make time-based
decisions in your business logic
OneSeedPublicSale.sol:763:29: Error: Avoid to make time-based
decisions in your business logic
OneSeedPublicSale.sol:775:18: Error: Avoid to make time-based
decisions in your business logic
OneSeedPublicSale.sol:775:99: Error: Use double quotes for string
literals
OneSeedPublicSale.sol:777:28: Error: Avoid to make time-based
decisions in your business logic
```

Software analysis result:

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