TOKEN AUDIT

CleanOcean

Smart Contract Security Audit

Audit Report

May, 2021

Table of contents

Project Introduction

Clean Ocean is a frictionless decentralized yield-generation charity eco-token. The token operates on an automated liquidity-locking and self-staking direct distribution protocol, providing safe, secure and hassle-free transactions and yield-generation for all holders automatically.

Name	CLEANOCEAN
Total Supply	1,000,000,000,000,000
Type	BSC Token
Website	https://cleanocean.io/
Platform	Ethereum / Solidity
Pancake Swap Supply	310,000,000,000,000
Holders	17,829 addresses
Deployed Contract	0x579F11C75Eb4e47F5290122e87CA411644aDCD97

Token Audit Team performed a security audit for Clean Ocean smart contracts during the period of Sep 3, 2021 to Sep 4, 2021.

The code for the audit was taken from following the official link:

https://github.com/eic0/CleanOcean/blob/main/CleanOcean.sol

Auditing Methodologies applied:

- In this audit, we can review the code listed below.
- The overall quality of code.
- Whether the implementation of ERC 20 standards.
- Whether the code is secure.
- Gas Optimization
- Code is safe from reentrancy and other vulnerabilities

Manual Audit

- Manually analyzing the source code line-by-line in an attempt to identify security vulnerabilities.
- Gas Consumption and optimization
- Assessing the overall project structure, complexity & quality.
- Checking whether all the libraries used in the code of the latest version.

Automated Audit

- Projects can be Automated using these tools with Slither, Manticore, Sol Graph others.
- Performing Unit testing.
- Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

➔ Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Auditing Tools

Language: Solidity

Platform and tools: Slither, Manti-Core, VScode, Solhint, Solc-select, Solidity-coverage

Audit Aim: The focus of the audit was to verify whether the smart contract is secure, resilient, and working according to the standard specs. The audit activity can be categories into three types

- Security
- Sound Architecture
- Code Correctness and Quality

Tokenomics:

The Tokenomics were as follows (in number of tokens):

1,000,000,000,000,000 Total Supply

300,000,000,000,000 Initial Burn

300,000,000,000,000 DxSale Pre-Sale Supply

310,000,000,000,000 PancakeSwap Supply

90,000,000,000,000 Dev-Wallet

(Numbers might vary slightly) 1,000,000,000,000,000 Total Supply

The founders hold around 8.5% of the total supply in a locked wallet.

This wallet exists to ensure the ongoing of the project and consists of the following parts:

50% Charity funds

30% Marketing budget

10% Development budget

10% Reserve

Issues Checking

We have scanned this smart contract code for commonly known and more specific Vulnerabilities that are below listed:

SN	Issue Description	Status
1	Re-entrancy	Verified
2	Compiler errors	Verified
3	Timestamp Dependence	Verified
4	Unsafe external calls	Verified
5	Gas Limit and Loops	Verified
6	DoS with Block Gas Limit	Verified
7	Private user data leaks	Verified
8	Code clones, functionality duplication	Verified
9	Style guide violation	Verified
10	Costly Loop	Verified
11	Balance equality	Verified
12	Unchecked math	Verified

13	Integer overflow/underflow	Verified
14	Cross-function Race Condition	Verified
15	Fallback function security	Verified
16	Data Consistency	Verified
17	Balance equality	Verified
18	ERC20 API violation	Verified
19	Deployment Consistency	Verified
20	Arithmetic accuracy	Verified
21	Transaction-Ordering Dependence (TOD) / Front Running	Verified
22	Address hardcoded	Verified
23	Scoping and Declarations	Verified
24	Implicit visibility level	Verified
25	Call Depth Attack (deprecated)	Verified

Severity Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	The issue affects the ability of the contract to compile or operate in a significant way.
Medium	Issues on this level could potentially bring problems and should eventually be fixed.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.
Informational	The issue has no impact on the contract's ability to operate.

Issues Found

- Critical severity issues - No Critical severity issues found.
- High severity issues - No Critical severity issues found.
- Medium severity issues - 1 medium severity issues found.
- Low severity issues - 1 low severity issues were found.
- Informational - 5 Informational severity issues were found.

High	Medium	Low	Informational
0	1	1	5

Medium level severity issues

1. Unused-Return

Severity: Medium

Description

The return value of an external call is not stored in a local or state variable:

```
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
    );
}
```

ignores return value

Suggestion:

Ensure that all the return values of the function calls are used.

Low level severity issues

1. block.timestamp

Severity: Low

Description

Uses timestamp for comparisons.
In the case of block.timestamp, developers often attempt to use it to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set a timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future

```
    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have permission to unlock");
        require(now > _lockTime , "Contract is locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
}
```

Suggestion:

Developers should write smart contracts with the notion that block values are not precise , and the use of them can lead to unexpected effects

https://consensys.github.io/smart-contract-best-practices/recommendations/#timestamp-dependence

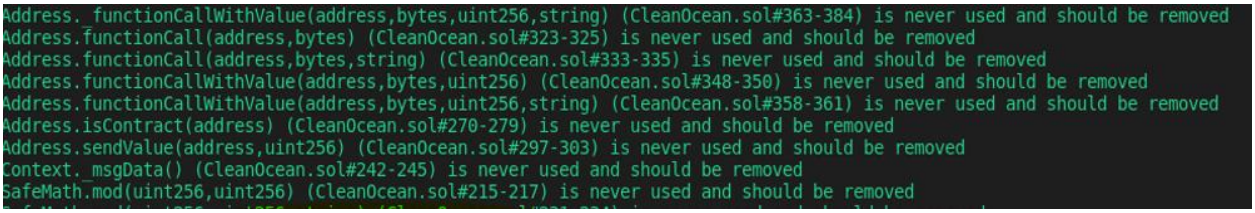
Informational level severity issues

1. DEAD-CODE

   Severity: Informational

   Description:

   Functions that are not sued. is not used in the contract, and make the code's review more difficult.



   Suggestion:

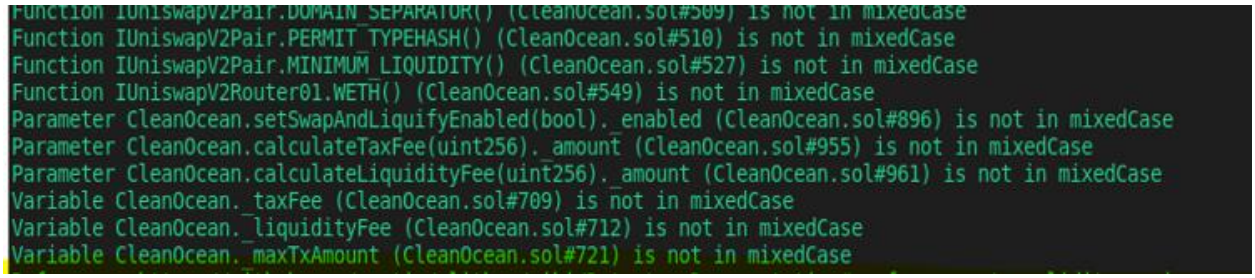
   Remove unused functions.

2. Conformance-To-Solidity-Naming-Conventions

   Severity: Informational

   Description:

   In the contract, many function names were found to be starting with capital letters. Functions other than constructors should use mixed Case



   Rule exceptions:

   Allow constant variable name/symbol/decimals to be lowercase (ERC20).
   Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

   Suggestion:

   Follow the Solidity: https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

3. State Variable Default Visibility

Severity: Informational

```
bool inSwapAndLiquify;
```

Description:

The Visibility of the inSwapAndLiquify variable is not defined. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.
The default is internal for state variables, but it should be made explicit.

Suggestion:

We recommend adding the visibility for the state variable of inSwapAndLiquify. Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

4. Public function that could be declared external

Severity: Informational

Description:

The following public functions that are never called by the contract should be declared external to save gas

- includeInFee
- setLiquidityFeePercent
- setSwapAndLiquifyEnabled
- excludeFromFee
- _reflectFee

Suggestion:

Use the external attribute for functions never called from the contract.

5. State variables that could be declared constant

Severity: Informational

Description:

The above constant state variables should be declared constant to save gas.

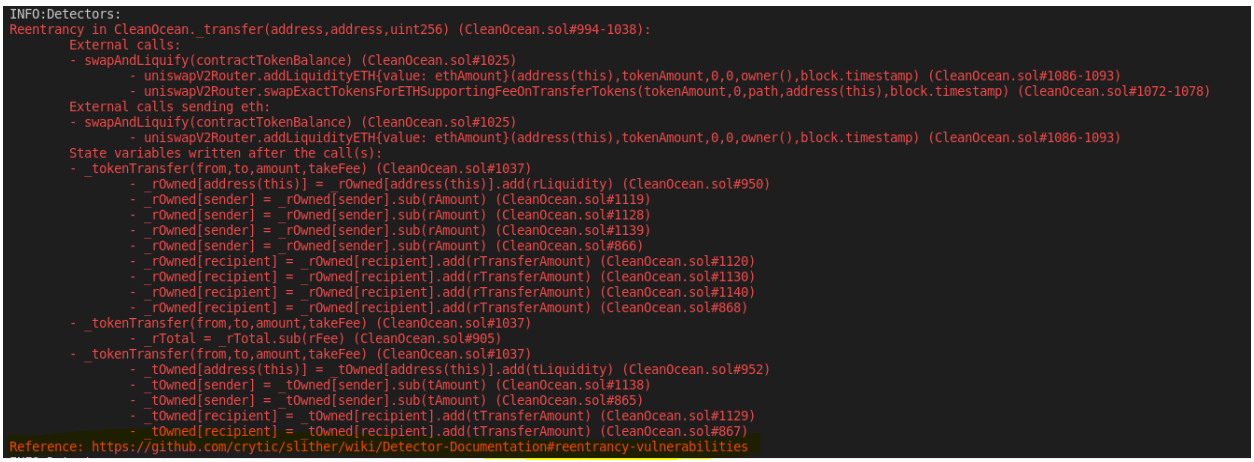
_decimals
_name
_symbol
_tTotal
numTokensSellToAddToLiqudity

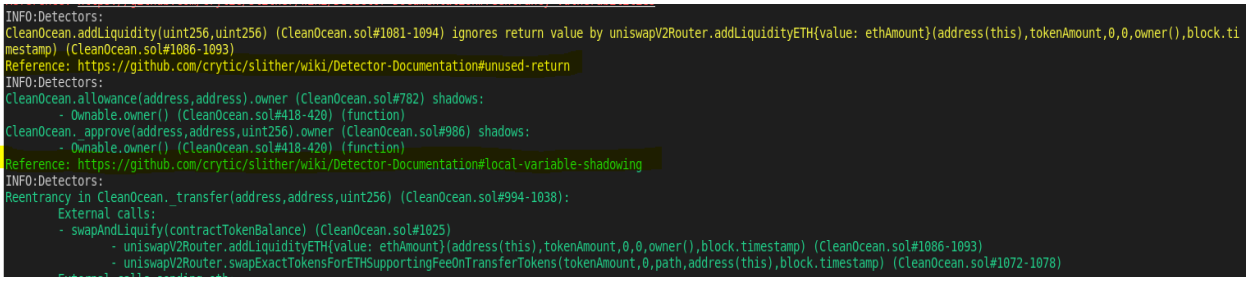
Suggestion:

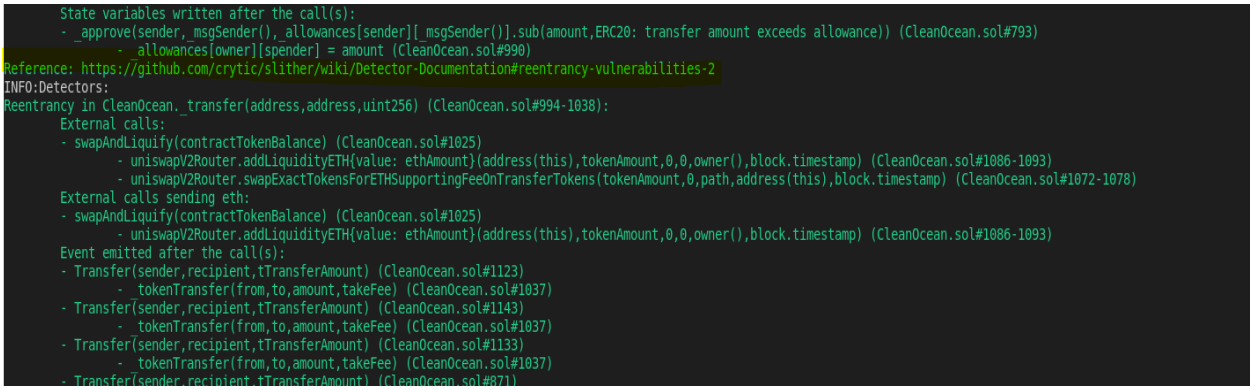
Add the constant attributes to state variables that never change

Automated Testing

We have to use Automated testing Slither. It is an Automated Analysis Tool in Smart Contract.



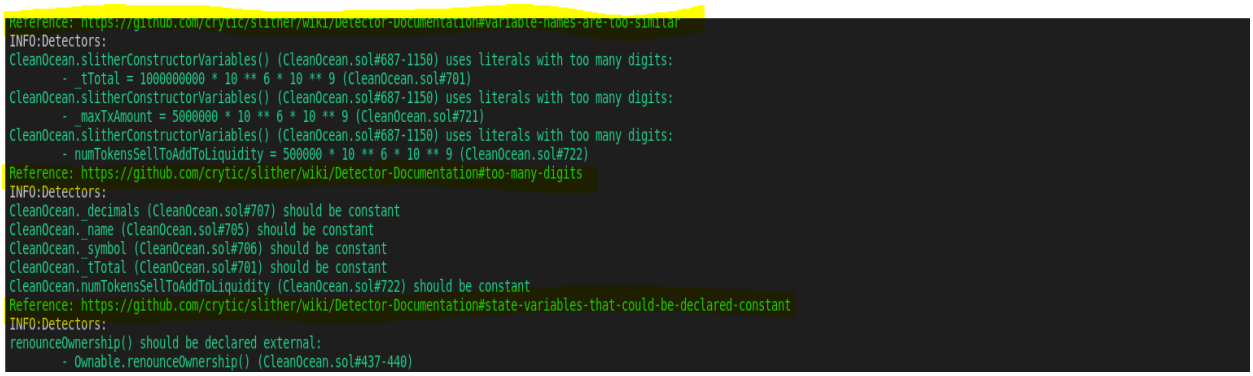


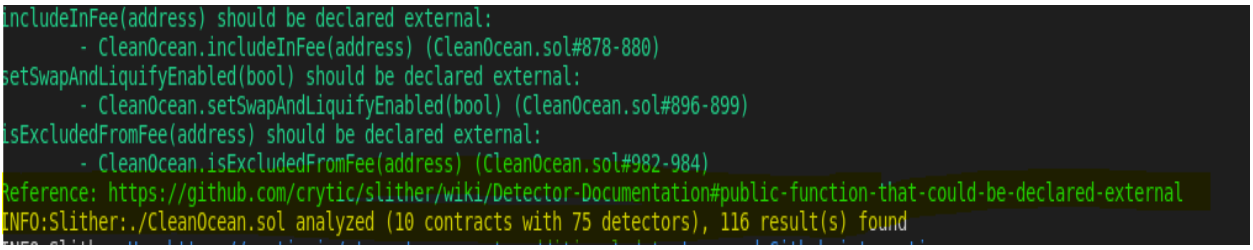


```
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (CleanOcean.sol#991)
                -  approve(sender, msgSender(), allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (CleanOcean.sol#793)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (CleanOcean.sol#465-470) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now >  lockTime,Contract is locked until 7 days) (CleanOcean.sol#467)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (CleanOcean.sol#270-279) uses assembly
        - INLINE ASM (CleanOcean.sol#277)
Address._functionCallWithValue(address,bytes,uint256,string) (CleanOcean.sol#363-384) uses assembly
        - INLINE ASM (CleanOcean.sol#376-379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (CleanOcean.sol#363-384) is never used and should be removed
Address.functionCall(address,bytes) (CleanOcean.sol#323-325) is never used and should be removed
Address.functionCall(address,bytes,string) (CleanOcean.sol#333-335) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CleanOcean.sol#348-350) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (CleanOcean.sol#358-361) is never used and should be removed
Address.isContract(address) (CleanOcean.sol#270-279) is never used and should be removed
Address.sendValue(address,uint256) (CleanOcean.sol#297-303) is never used and should be removed
Context._msgData() (CleanOcean.sol#242-245) is never used and should be removed
SafeMath.mod(uint256,uint256) (CleanOcean.sol#215-217) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CleanOcean.sol#231-234) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
CleanOcean._rTotal (CleanOcean.sol#702) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX %  tTotal))
CleanOcean._previousTaxFee (CleanOcean.sol#710) is set pre-construction with a non-constant function or state variable:
        -  _taxFee
CleanOcean._previousLiquidityFee (CleanOcean.sol#713) is set pre-construction with a non-constant function or state variable:
        -  liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (CleanOcean.sol#297-303):
        - (success) = recipient.call{value: amount}() (CleanOcean.sol#301)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (CleanOcean.sol#363-384):
        - (success,returndata) = target.call{value: weiValue}(data) (CleanOcean.sol#367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (CleanOcean.sol#509) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (CleanOcean.sol#510) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (CleanOcean.sol#527) is not in mixedCase
Function IUniswapV2Router01.WETH() (CleanOcean.sol#549) is not in mixedCase
Parameter CleanOcean.setSwapAndLiquifyEnabled(bool)._enabled (CleanOcean.sol#896) is not in mixedCase
Parameter CleanOcean.calculateTaxFee(uint256)._amount (CleanOcean.sol#955) is not in mixedCase
Parameter CleanOcean.calculateLiquidityFee(uint256)._amount (CleanOcean.sol#961) is not in mixedCase
Variable CleanOcean._taxFee (CleanOcean.sol#709) is not in mixedCase
Variable CleanOcean._liquidityFee (CleanOcean.sol#712) is not in mixedCase
Variable CleanOcean._maxTxAmount (CleanOcean.sol#721) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (CleanOcean.sol#243)" inContext (CleanOcean.sol#237-246)
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 (CleanOcean.sol#554) is too similar to IUniswapV2Router01.addLiquid
ity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (CleanOcean.sol#555)
Variable CleanOcean._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (CleanOcean.sol#926) is too similar to CleanOcean._transferBothExcluded(address,address,uint256).tTransferA
mount (CleanOcean.sol#864)
Variable CleanOcean._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (CleanOcean.sol#926) is too similar to CleanOcean._transferToExcluded(address,address,uint256).tTransferAmo
unt (CleanOcean.sol#1127)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
        -  tTotal = 1000000000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#701)
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
        -  maxTxAmount = 5000000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#721)
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 500000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#722)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
CleanOcean._decimals (CleanOcean.sol#707) should be constant
CleanOcean._name (CleanOcean.sol#705) should be constant
CleanOcean._symbol (CleanOcean.sol#706) should be constant
CleanOcean._tTotal (CleanOcean.sol#701) should be constant
CleanOcean.numTokensSellToAddToLiquidity (CleanOcean.sol#722) 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() (CleanOcean.sol#437-440)
```

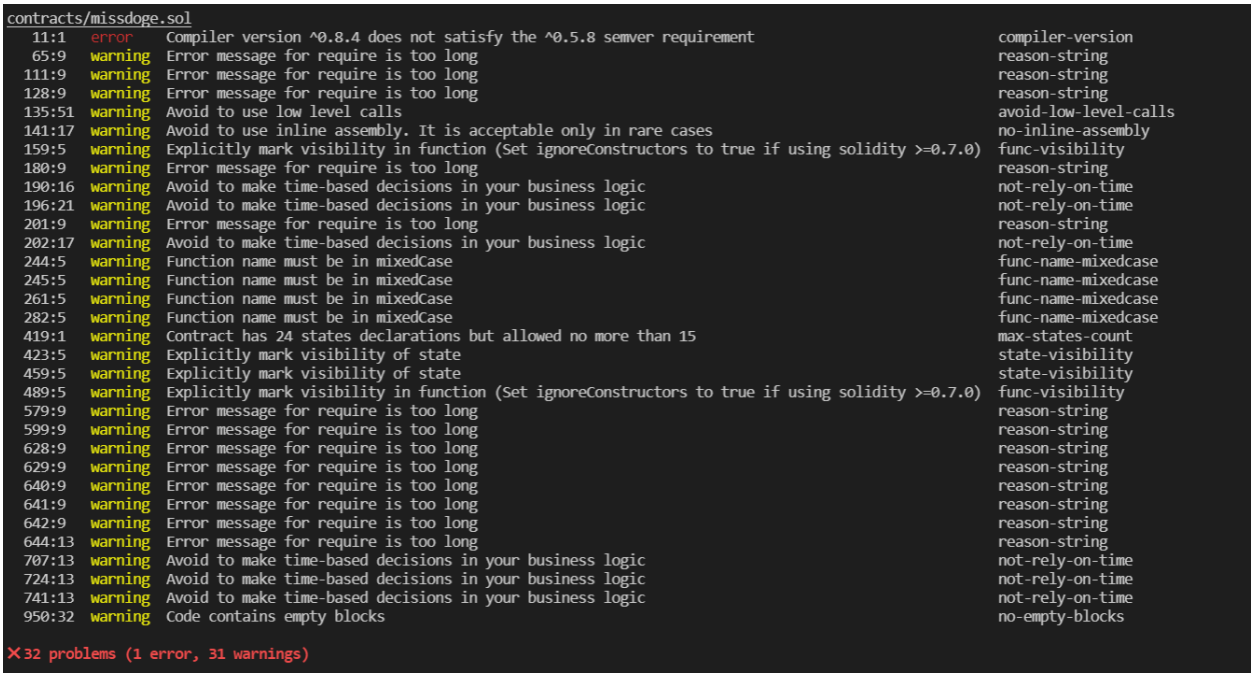

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
- _tTotal = 1000000000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#701)
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
- _maxTxAmount = 5000000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#721)
CleanOcean.slitherConstructorVariables() (CleanOcean.sol#687-1150) uses literals with too many digits:
- numTokensSellToAddToLiquidity = 500000 * 10 ** 6 * 10 ** 9 (CleanOcean.sol#722)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
CleanOcean._decimals (CleanOcean.sol#707) should be constant
CleanOcean._name (CleanOcean.sol#705) should be constant
CleanOcean._symbol (CleanOcean.sol#706) should be constant
CleanOcean._tTotal (CleanOcean.sol#701) should be constant
CleanOcean.numTokensSellToAddToLiquidity (CleanOcean.sol#722) 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() (CleanOcean.sol#437-440)


includeInFee(address) should be declared external:
- CleanOcean.includeInFee(address) (CleanOcean.sol#878-880)
setSwapAndLiquifyEnabled(bool) should be declared external:
- CleanOcean.setSwapAndLiquifyEnabled(bool) (CleanOcean.sol#896-899)
isExcludedFromFee(address) should be declared external:
- CleanOcean.isExcludedFromFee(address) (CleanOcean.sol#982-984)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./CleanOcean.sol analyzed (10 contracts with 75 detectors), 116 result(s) found

Sol-hint Tool:

A linter for Solidity that provides both Security and Style Guide validations.

Coding style issues influence code readability and, in some cases, may lead to bugs in future. Smart Contracts have a naming convention, indentation and code layout issues. It's recommended to use Solidity Style Guide to fix all the issues. Consider following the Solidity guidelines on formatting the code and commenting for all the files. It can improve the overall code quality and readability



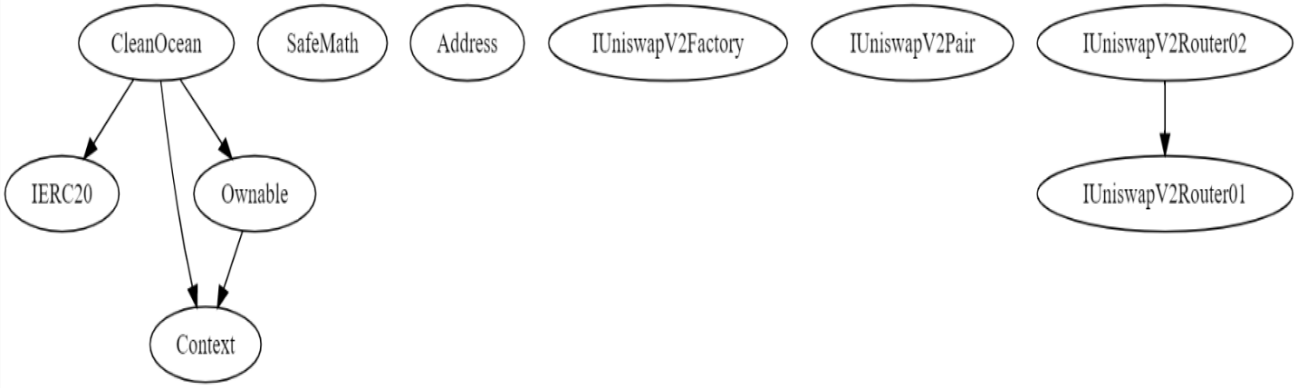
```
contracts/missdoge.sol
   11:1   error    Compiler version ^0.8.4 does not satisfy the ^0.5.8 semver requirement                    compiler-version
   65:9   warning  Error message for require is too long                                                    reason-string
  111:9   warning  Error message for require is too long                                                    reason-string
  128:9   warning  Error message for require is too long                                                    reason-string
  135:51  warning  Avoid to use low level calls                                                             avoid-low-level-calls
  141:17  warning  Avoid to use inline assembly. It is acceptable only in rare cases                        no-inline-assembly
  159:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  180:9   warning  Error message for require is too long                                                    reason-string
  190:16  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  196:21  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  201:9   warning  Error message for require is too long                                                    reason-string
  202:17  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  244:5   warning  Function name must be in mixedCase                                                      func-name-mixedcase
  245:5   warning  Function name must be in mixedCase                                                      func-name-mixedcase
  261:5   warning  Function name must be in mixedCase                                                      func-name-mixedcase
  282:5   warning  Function name must be in mixedCase                                                      func-name-mixedcase
  419:1   warning  Contract has 24 states declarations but allowed no more than 15                        max-states-count
  423:5   warning  Explicitly mark visibility of state                                                     state-visibility
  459:5   warning  Explicitly mark visibility of state                                                     state-visibility
  489:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  579:9   warning  Error message for require is too long                                                    reason-string
  599:9   warning  Error message for require is too long                                                    reason-string
  628:9   warning  Error message for require is too long                                                    reason-string
  629:9   warning  Error message for require is too long                                                    reason-string
  640:9   warning  Error message for require is too long                                                    reason-string
  641:9   warning  Error message for require is too long                                                    reason-string
  642:9   warning  Error message for require is too long                                                    reason-string
  644:13  warning  Error message for require is too long                                                    reason-string
  707:13  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  724:13  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  741:13  warning  Avoid to make time-based decisions in your business logic                                not-rely-on-time
  950:32  warning  Code contains empty blocks                                                               no-empty-blocks

✖32 problems (1 error, 31 warnings)
```

Functional View

Function	Return Type	Test Result
name	Public	Verified
symbol	Public	Verified
decimals	Public	Verified
totalSupply	Public	Verified
balanceOf	Public	Verified
transfer	Public	Verified
allowance	Public	Verified
approve	Public	Verified
TransferFrom	Public	Verified
increaseAllowance	Public	Verified
decreaseAllowance	Public	Verified
isExcludedFromReward	Public	Verified
totalFees	Public	Verified
deliver	Public	Verified
reflectionFromToken	Public	Verified

tokenFromReflection	Public	Verified
excludeFromReward	Public	Verified
includeInReward	Public	Verified
transferBothExcluded	Public	Verified

Inheritance Chart



Audit Findings Results

There were 1 Medium and 1 Low issue and 5 Informational found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner. None of the critical issues were resolved.

Generally, the contracts are well written and structured. The findings during the audit have some impact on contract performance or security

Disclaimer

This audit does not provide a security or correctness guarantee of audited smart contracts You agree that your access and/or use, including but not limited to any services, products, platforms, content, will be at your Own risk. Smart contract remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security.


http://tokenaudit.net/


https://t.me/TokenAuditt


https://twitter.com/AuditToken