Xiasi Inu
Smart Contract Security Audit

Audit Report
Aug, 2021

Table of contents

Project Introduction

The Xiasi Dog is respected by people due to the belief that the dog brings wealth to a family. The team behind Xiasi also aims to be well respected by its holders due to our openness, transparency and rewarding tokenomics. We are committed to not only build innovative utility (Xiasi merch, swap and IDO platform) but to give back wealth to the families that are most in need with charitable donations. The team strives to build a valued and thriving community of holders united behind our vision whilst engaging, building partnerships and providing the "fun factor" along the way.

Name	Xiasi Inu (XIASI)
Website	https://xiasi.finance/
Total Supply	600,000,000,000,000 [1 Quadrillion]
Type	BSC Token
Platform	Ethereum / Solidity
Holders	10,720 addresses
Deployed Contract	0x0e20E3216EA172fcf9eAa19723b119e090fD353f

Token Audit Team performed a security audit for Xiasi Inu smart contracts during the period of Aug 16, 2021 to Aug 16, 2021.

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

https://github.com/InuXiasi/Xiasi-Inu-Contracts/blob/main/Xiasi.sol

Auditing Methodologies applied:

- In this audit, we consider the following crucial features of the code.
- The overall quality of code.
- Whether the implementation of ERC 20/BEP 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, SolGraph 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 Quadrillion Total Supply

8% Redistribution

4% Rewarded to holders

4% Added to Liquidity Pool

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	-	1 Critical severity issues found.
● Medium severity issues	-	1 medium severity issues found.
● Low severity issues	-	2 low severity issues were found.
● Informational	-	5 Informational severity issues were found

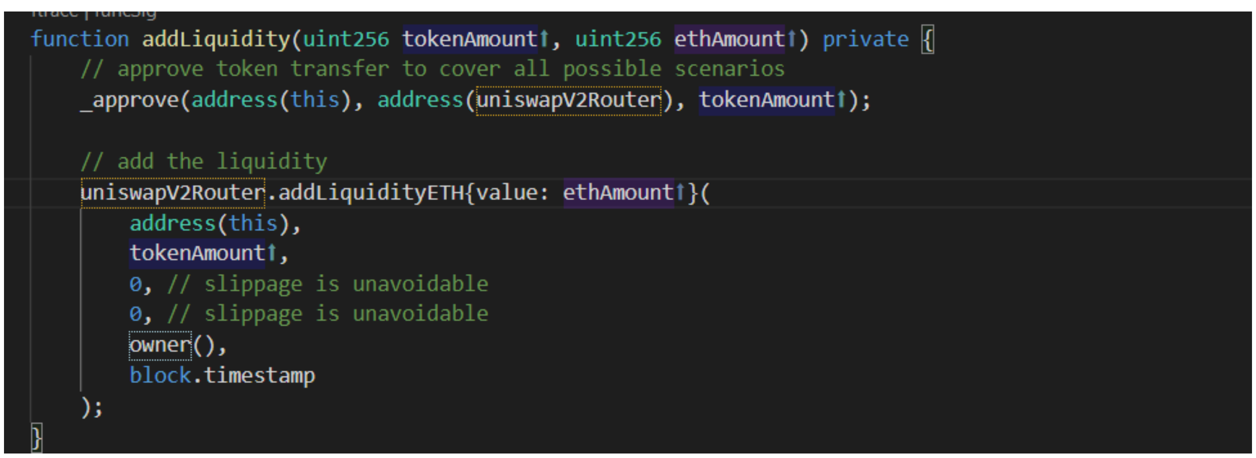
High	Medium	Low	Informational
1	1	2	5

High level severity issues

1. Functions that send Ether to arbitrary destinations

   Severity: High

   Description: Unprotected call to a function sending Ether to an arbitrary address.



```
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
    );
}
```

Suggestion:

Ensure that an arbitrary user cannot withdraw unauthorized funds.

TOKEN AUDIT

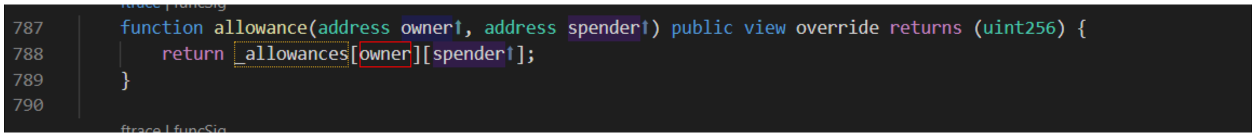
Medium level severity issues

1. Reentrancy Attack

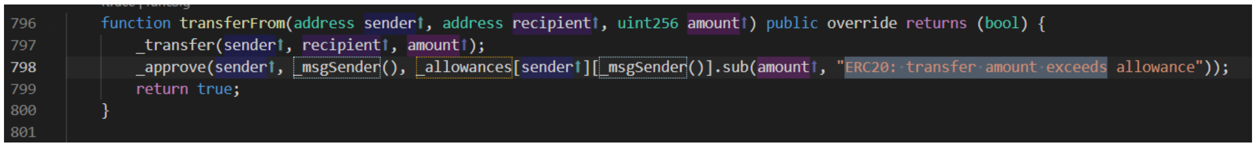
   Severity: Medium

   Description: Calling Xiasi.transfer() and Xiasi.transferFrom() might trigger function uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTo kens() and uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTo kens() , which is implemented by a third party at uniswapV2Router. If there are vulnerable external calls in uniswapV2Router, reentrancy attacks could be conducted because these two functions have state updates and event emits after external calls.

   The scope of the audit would treat the third-party implementation at uniswapV2Router as a black box and assume its functional correctness. However, third parties may be compromised in the real world that leads to assets lost or stolen.



```
    function allowance(address owner, address spender) public view override returns (uint256) {
        return _allowances[owner][spender];
    }

```



```
    function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
        return true;
    }

```

   Remediation:

   We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attacks.

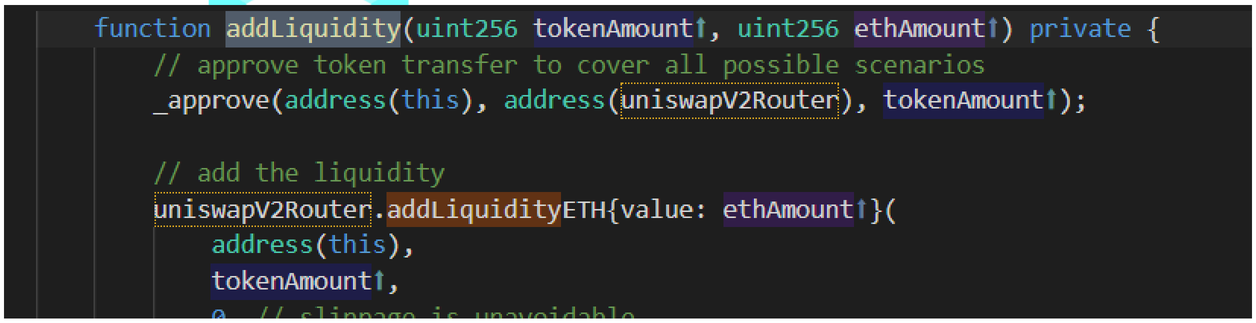
Low level severity issues

1. Unused code (or) Dead code

Severity: Low

Description:

no state changes and has no side effects that alter data or control flow, such that removal of the code would have no impact on functionality or correctness



```
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
```

Remediation:

The program contains code that is not essential for execution, i.e., makes no state changes and has no side effects that alter data or control flow

2. State variables that could be declared constant

Severity: Low

Description:

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

- _name
- _symbol
- _decimals
- _tTotal

TOKEN AUDIT

Remediation:

Add the constant attributes to state variables that never change

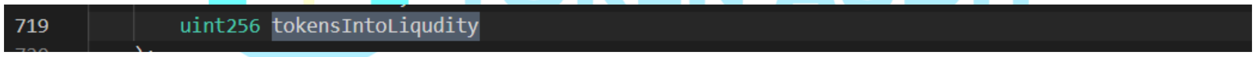
Informational level severity issues

1. Variable Typos

Severity: Informational

Description:

There are typos in the above variables



```
uint256 tokensIntoLiqudity
```

Remediation:

We recommend correcting and changing tokensIntoLiqudity to tokensIntoLiquidity

2. 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 mixedCase

Examples: getBalance, transfer, verifyOwner

Remediation:

Solidity naming convention

3. Public function that could be declared external

Severity: Informational

Description:

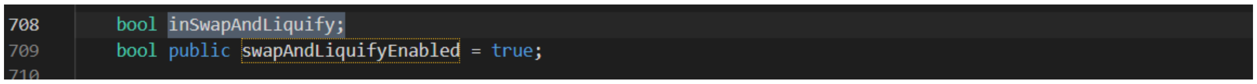
The following public functions that are never called by the contract should be declared external to save gas:

- totalFees()
- deliver()
- reflectionFromToken()
- tokenFromReflection()
- excludeFromReward()

Remediation:
Use the external attribute for functions never called from the contract

4. State Variable Default Visibility

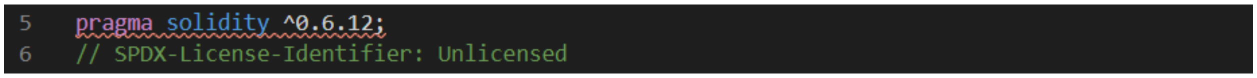
Severity: Informational
Description:
The default is internal for state variables, but it should be made explicit


```
bool inSwapAndLiquify;
bool public swapAndLiquifyEnabled = true;
```

Remediation:
We recommend adding the visibility for the state variable of inSwapAndLiquify.

5. pragma and Incorrect versions of Solidity

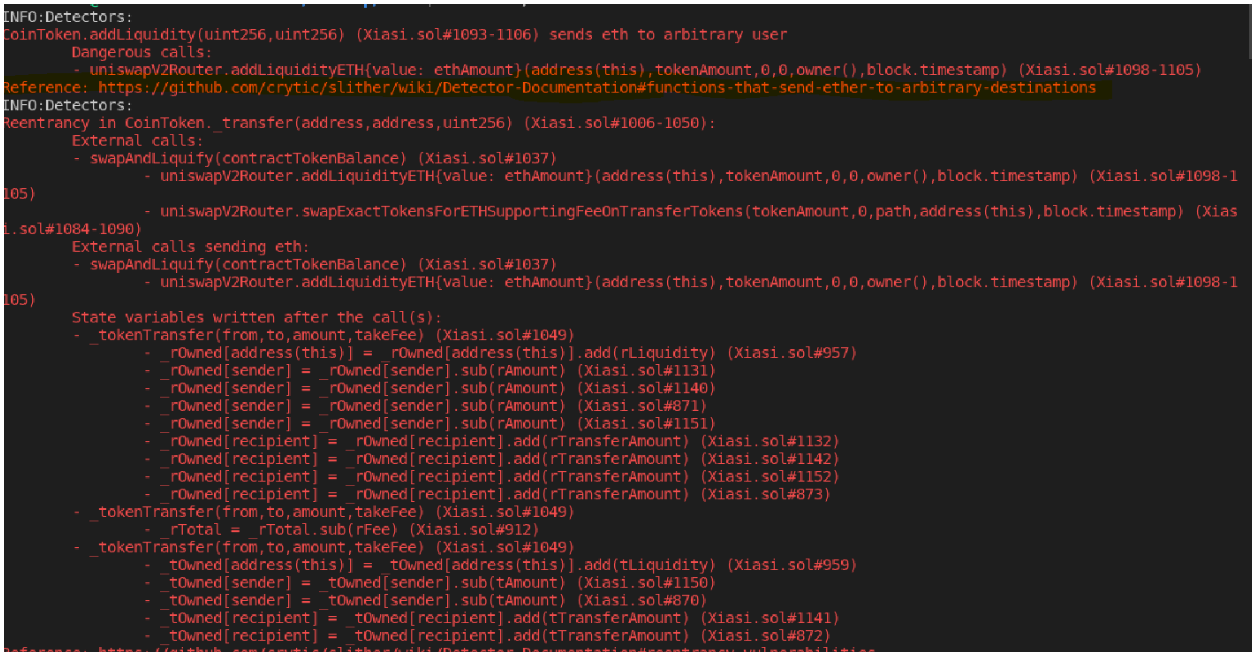
Severity: Informational
Description:
Solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statements


```
pragma solidity ^0.6.12;
// SPDX-License-Identifier: Unlicensed
```

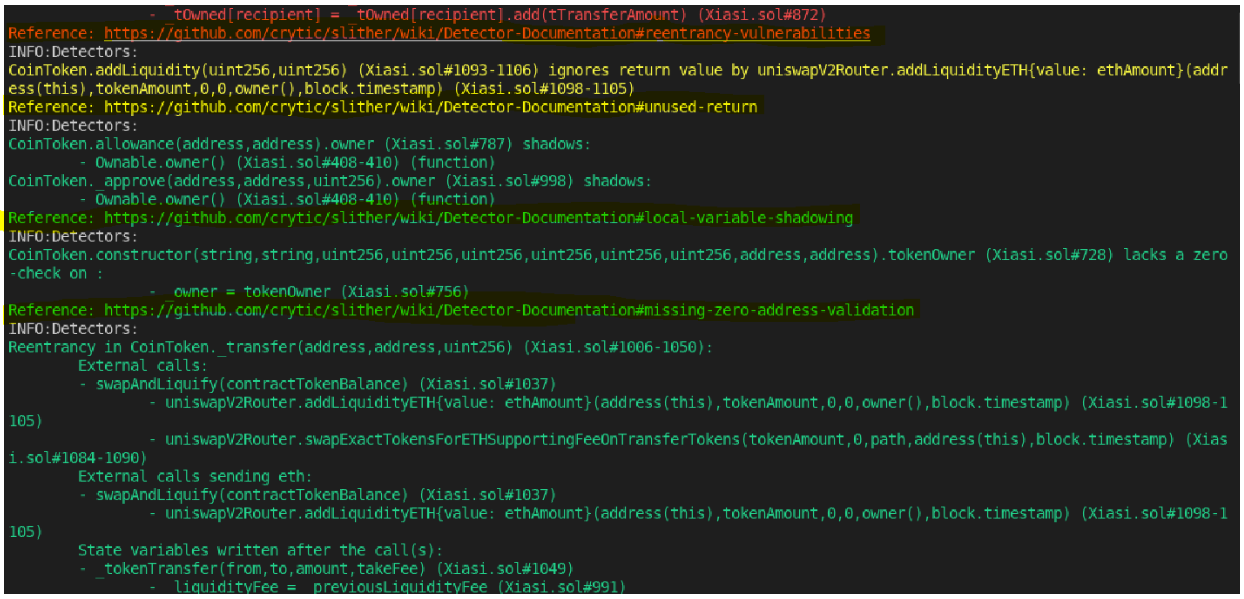
Remediation:
Compiles the latest version 0.8.4 only That we can find Errors easily.

Automated Testing

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

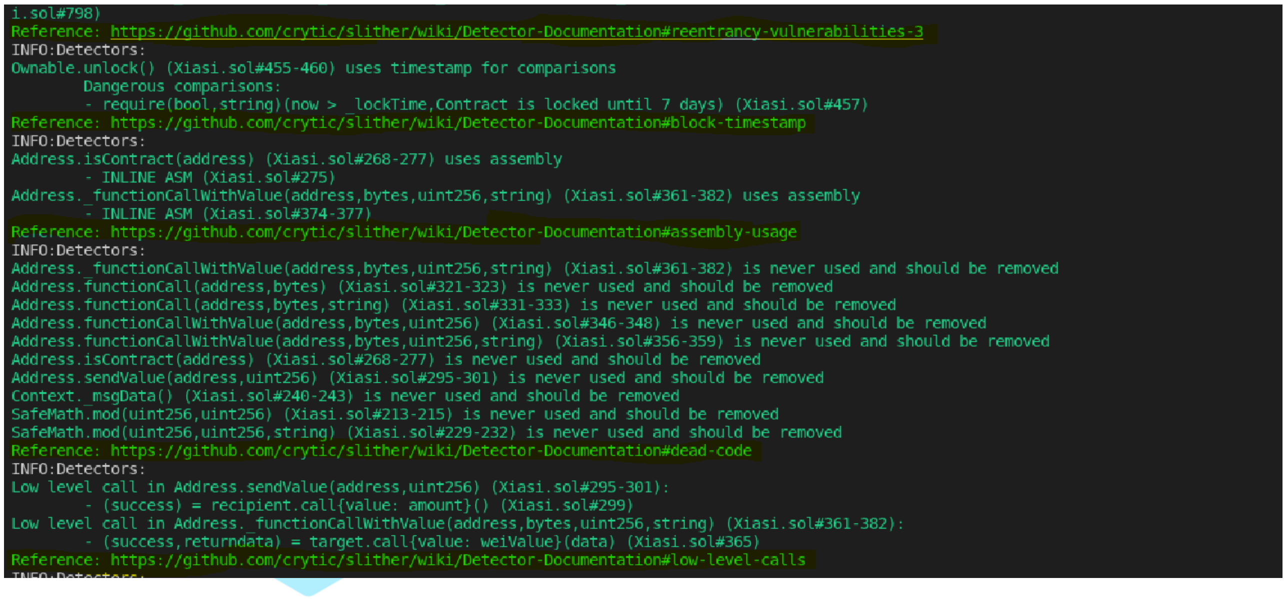


```
INFO:Detectors:
CoinToken.addLiquidity(uint256,uint256) (Xiasi.sol#1093-1106) sends eth to arbitrary user
        Dangerous calls:
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in CoinToken._transfer(address,address,uint256) (Xiasi.sol#1006-1050):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Xiasi.sol#1037)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1
105)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Xias
i.sol#1084-1090)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Xiasi.sol#1037)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1
105)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (Xiasi.sol#1049)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (Xiasi.sol#957)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Xiasi.sol#1131)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Xiasi.sol#1140)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Xiasi.sol#871)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (Xiasi.sol#1151)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Xiasi.sol#1132)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Xiasi.sol#1142)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Xiasi.sol#1152)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Xiasi.sol#873)
        - _tokenTransfer(from,to,amount,takeFee) (Xiasi.sol#1049)
                - _rTotal = _rTotal.sub(rFee) (Xiasi.sol#912)
        - _tokenTransfer(from,to,amount,takeFee) (Xiasi.sol#1049)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (Xiasi.sol#959)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Xiasi.sol#1150)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (Xiasi.sol#870)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Xiasi.sol#1141)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Xiasi.sol#872)
```

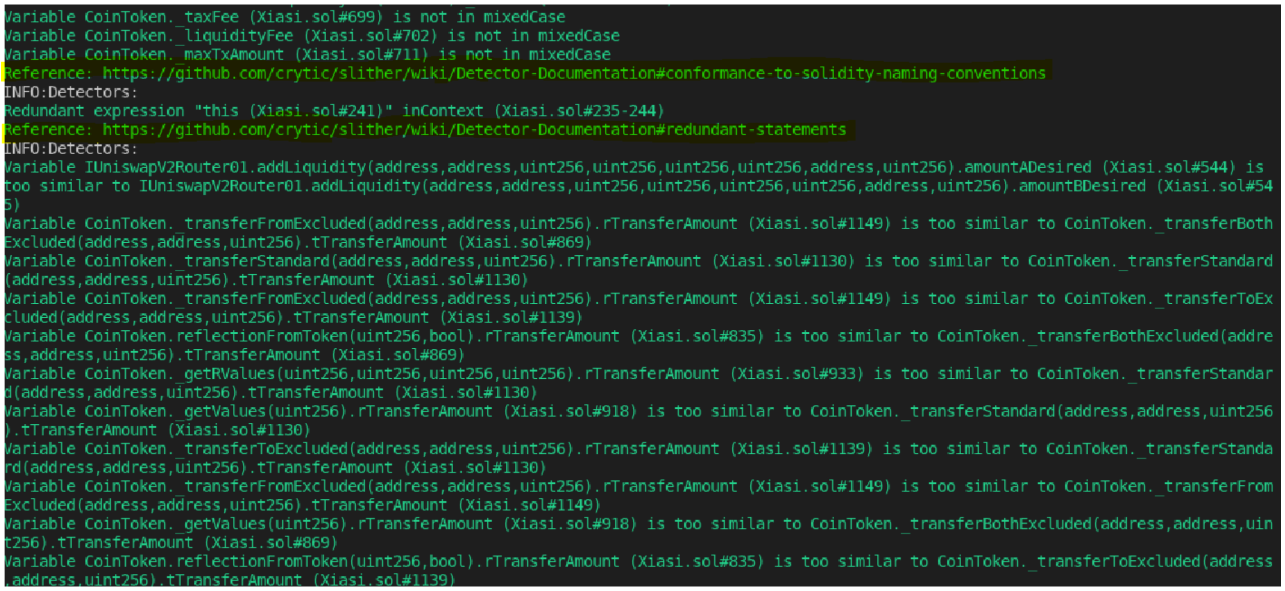


```
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Xiasi.sol#872)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
CoinToken.addLiquidity(uint256,uint256) (Xiasi.sol#1093-1106) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(addr
ess(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CoinToken.allowance(address,address).owner (Xiasi.sol#787) shadows:
        - Ownable.owner() (Xiasi.sol#408-410) (function)
CoinToken._approve(address,address,uint256).owner (Xiasi.sol#998) shadows:
        - Ownable.owner() (Xiasi.sol#408-410) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CoinToken.constructor(string,string,uint256,uint256,uint256,uint256,uint256,uint256,address,address).tokenOwner (Xiasi.sol#728) lacks a zero
-check on :
                - _owner = tokenOwner (Xiasi.sol#756)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CoinToken._transfer(address,address,uint256) (Xiasi.sol#1006-1050):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Xiasi.sol#1037)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1
105)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Xias
i.sol#1084-1090)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Xiasi.sol#1037)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Xiasi.sol#1098-1
105)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (Xiasi.sol#1049)
                - _liquidityFee = _previousLiquidityFee (Xiasi.sol#991)
```

TOKEN AUDIT

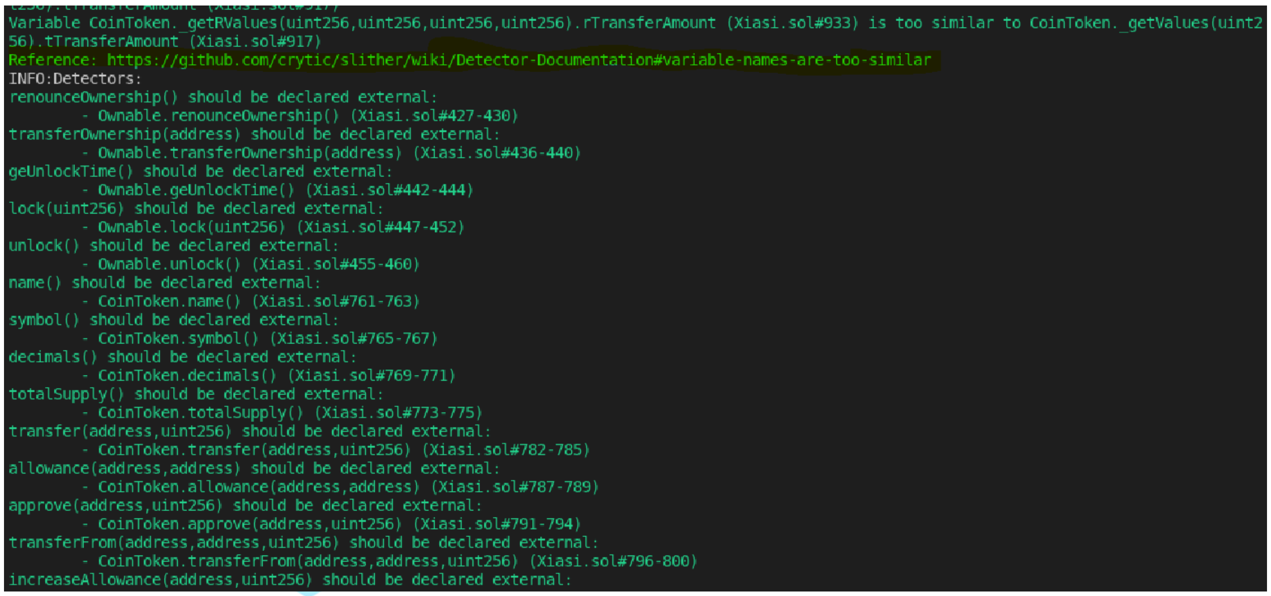


```
i.sol#798)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (Xiasi.sol#455-460) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked until 7 days) (Xiasi.sol#457)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Xiasi.sol#268-277) uses assembly
        - INLINE ASM (Xiasi.sol#275)
Address._functionCallWithValue(address,bytes,uint256,string) (Xiasi.sol#361-382) uses assembly
        - INLINE ASM (Xiasi.sol#374-377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (Xiasi.sol#361-382) is never used and should be removed
Address.functionCall(address,bytes) (Xiasi.sol#321-323) is never used and should be removed
Address.functionCall(address,bytes,string) (Xiasi.sol#331-333) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Xiasi.sol#346-348) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Xiasi.sol#356-359) is never used and should be removed
Address.isContract(address) (Xiasi.sol#268-277) is never used and should be removed
Address.sendValue(address,uint256) (Xiasi.sol#295-301) is never used and should be removed
Context._msgData() (Xiasi.sol#240-243) is never used and should be removed
SafeMath.mod(uint256,uint256) (Xiasi.sol#213-215) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Xiasi.sol#229-232) 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) (Xiasi.sol#295-301):
        - (success) = recipient.call{value: amount}() (Xiasi.sol#299)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Xiasi.sol#361-382):
        - (success,returndata) = target.call{value: weiValue}(data) (Xiasi.sol#365)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
```

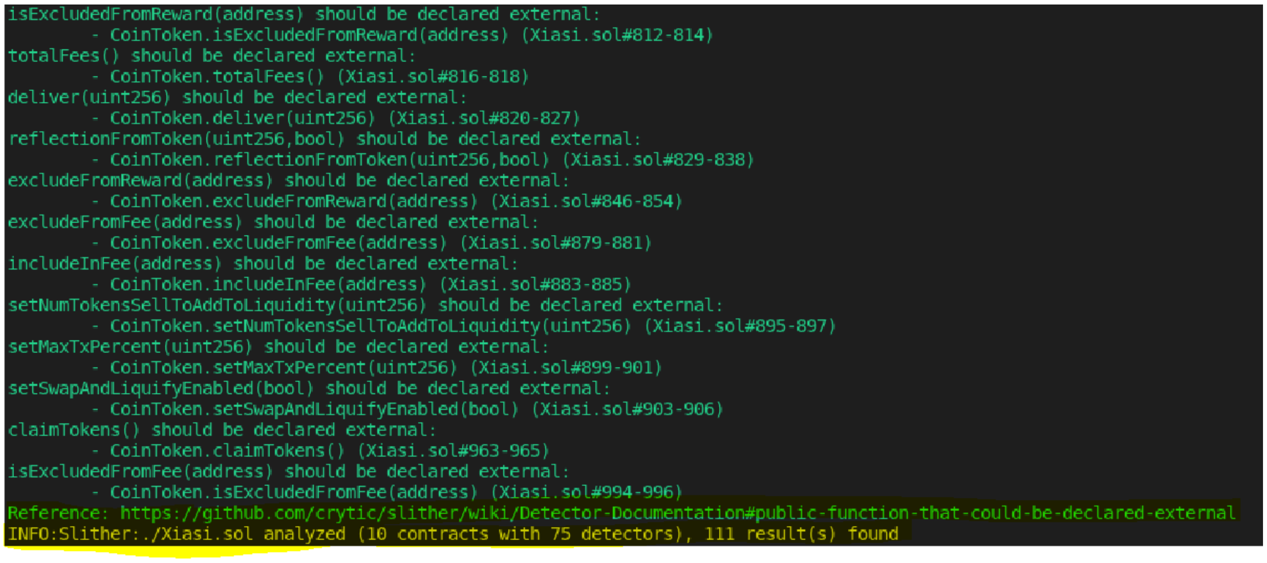


```
Variable CoinToken._taxFee (Xiasi.sol#699) is not in mixedCase
Variable CoinToken._liquidityFee (Xiasi.sol#702) is not in mixedCase
Variable CoinToken._maxTxAmount (Xiasi.sol#711) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Xiasi.sol#241)" inContext (Xiasi.sol#235-244)
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 (Xiasi.sol#544) is
too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Xiasi.sol#54
5)
Variable CoinToken._transferFromExcluded(address,address,uint256).rTransferAmount (Xiasi.sol#1149) is too similar to CoinToken._transferBoth
Excluded(address,address,uint256).tTransferAmount (Xiasi.sol#869)
Variable CoinToken._transferStandard(address,address,uint256).rTransferAmount (Xiasi.sol#1130) is too similar to CoinToken._transferStandard
(address,address,uint256).tTransferAmount (Xiasi.sol#1130)
Variable CoinToken._transferFromExcluded(address,address,uint256).rTransferAmount (Xiasi.sol#1149) is too similar to CoinToken._transferToEx
cluded(address,address,uint256).tTransferAmount (Xiasi.sol#1139)
Variable CoinToken.reflectionFromToken(uint256,bool).rTransferAmount (Xiasi.sol#835) is too similar to CoinToken._transferBothExcluded(addre
ss,address,uint256).tTransferAmount (Xiasi.sol#869)
Variable CoinToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Xiasi.sol#933) is too similar to CoinToken._transferStandar
d(address,address,uint256).tTransferAmount (Xiasi.sol#1130)
Variable CoinToken._getValues(uint256).rTransferAmount (Xiasi.sol#918) is too similar to CoinToken._transferStandard(address,address,uint256
).tTransferAmount (Xiasi.sol#1130)
Variable CoinToken._transferToExcluded(address,address,uint256).rTransferAmount (Xiasi.sol#1139) is too similar to CoinToken._transferStanda
rd(address,address,uint256).tTransferAmount (Xiasi.sol#1130)
Variable CoinToken._transferFromExcluded(address,address,uint256).rTransferAmount (Xiasi.sol#1149) is too similar to CoinToken._transferFrom
Excluded(address,address,uint256).tTransferAmount (Xiasi.sol#1149)
Variable CoinToken._getValues(uint256).rTransferAmount (Xiasi.sol#918) is too similar to CoinToken._transferBothExcluded(address,address,uin
t256).tTransferAmount (Xiasi.sol#869)
Variable CoinToken.reflectionFromToken(uint256,bool).rTransferAmount (Xiasi.sol#835) is too similar to CoinToken._transferToExcluded(address
,address,uint256).tTransferAmount (Xiasi.sol#1139)
```

TOKEN AUDIT



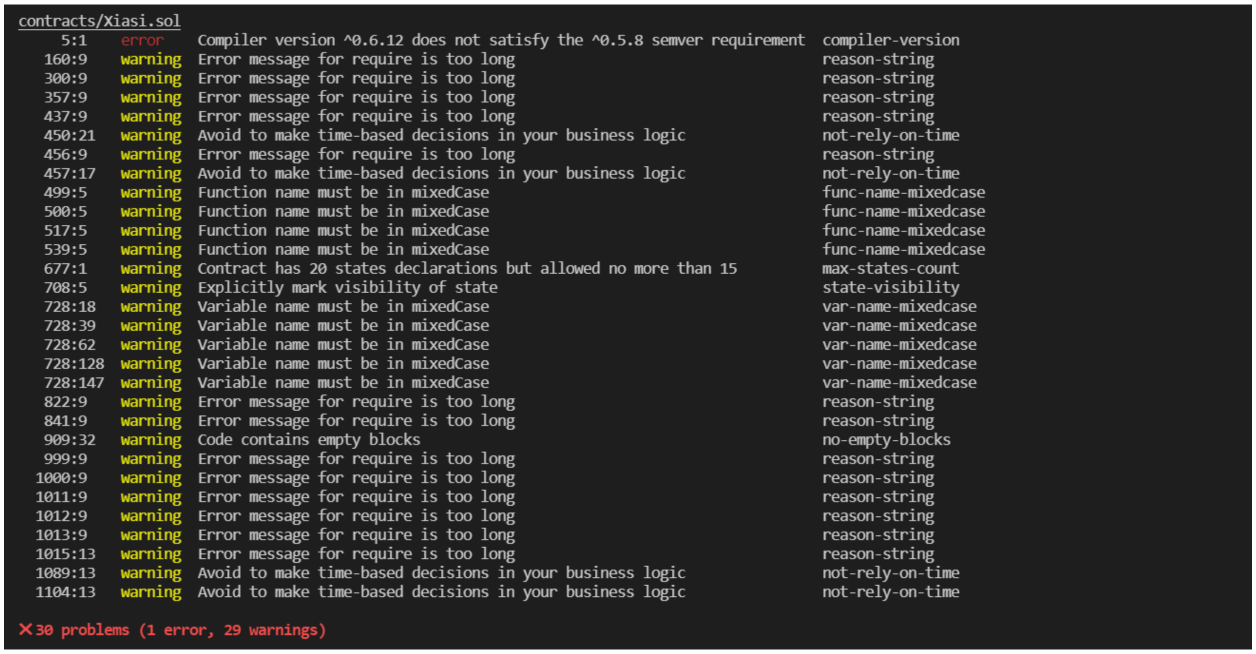
```
Variable CoinToken._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (Xiasi.sol#933) is too similar to CoinToken._getValues(uint2
56).tTransferAmount (Xiasi.sol#917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Xiasi.sol#427-430)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Xiasi.sol#436-440)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (Xiasi.sol#442-444)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Xiasi.sol#447-452)
unlock() should be declared external:
        - Ownable.unlock() (Xiasi.sol#455-460)
name() should be declared external:
        - CoinToken.name() (Xiasi.sol#761-763)
symbol() should be declared external:
        - CoinToken.symbol() (Xiasi.sol#765-767)
decimals() should be declared external:
        - CoinToken.decimals() (Xiasi.sol#769-771)
totalSupply() should be declared external:
        - CoinToken.totalSupply() (Xiasi.sol#773-775)
transfer(address,uint256) should be declared external:
        - CoinToken.transfer(address,uint256) (Xiasi.sol#782-785)
allowance(address,address) should be declared external:
        - CoinToken.allowance(address,address) (Xiasi.sol#787-789)
approve(address,uint256) should be declared external:
        - CoinToken.approve(address,uint256) (Xiasi.sol#791-794)
transferFrom(address,address,uint256) should be declared external:
        - CoinToken.transferFrom(address,address,uint256) (Xiasi.sol#796-800)
increaseAllowance(address,uint256) should be declared external:
```



```
isExcludedFromReward(address) should be declared external:
        - CoinToken.isExcludedFromReward(address) (Xiasi.sol#812-814)
totalFees() should be declared external:
        - CoinToken.totalFees() (Xiasi.sol#816-818)
deliver(uint256) should be declared external:
        - CoinToken.deliver(uint256) (Xiasi.sol#820-827)
reflectionFromToken(uint256,bool) should be declared external:
        - CoinToken.reflectionFromToken(uint256,bool) (Xiasi.sol#829-838)
excludeFromReward(address) should be declared external:
        - CoinToken.excludeFromReward(address) (Xiasi.sol#846-854)
excludeFromFee(address) should be declared external:
        - CoinToken.excludeFromFee(address) (Xiasi.sol#879-881)
includeInFee(address) should be declared external:
        - CoinToken.includeInFee(address) (Xiasi.sol#883-885)
setNumTokensSellToAddToLiquidity(uint256) should be declared external:
        - CoinToken.setNumTokensSellToAddToLiquidity(uint256) (Xiasi.sol#895-897)
setMaxTxPercent(uint256) should be declared external:
        - CoinToken.setMaxTxPercent(uint256) (Xiasi.sol#899-901)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - CoinToken.setSwapAndLiquifyEnabled(bool) (Xiasi.sol#903-906)
claimTokens() should be declared external:
        - CoinToken.claimTokens() (Xiasi.sol#963-965)
isExcludedFromFee(address) should be declared external:
        - CoinToken.isExcludedFromFee(address) (Xiasi.sol#994-996)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:./Xiasi.sol analyzed (10 contracts with 75 detectors), 111 result(s) found
```

Solhint Tool:

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



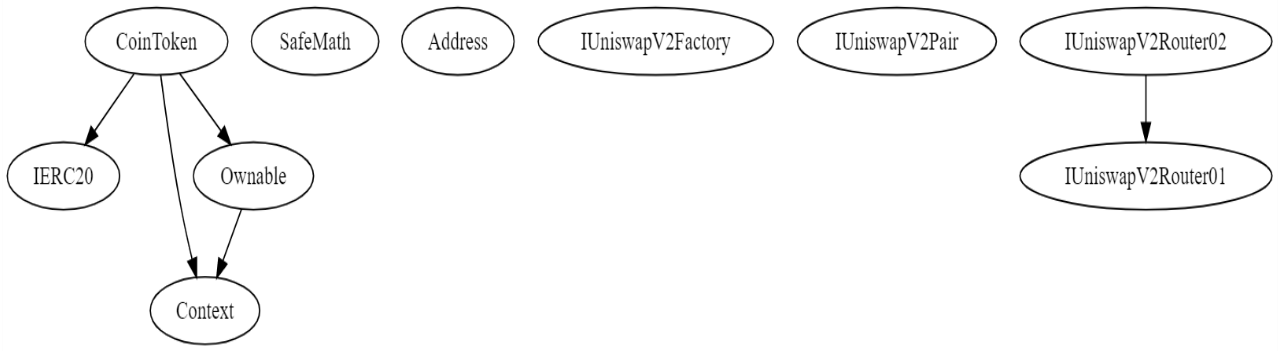
```
contracts/Xiasi.sol
     5:1    error    Compiler version ^0.6.12 does not satisfy the ^0.5.8 semver requirement  compiler-version
   160:9    warning  Error message for require is too long                                    reason-string
   300:9    warning  Error message for require is too long                                    reason-string
   357:9    warning  Error message for require is too long                                    reason-string
   437:9    warning  Error message for require is too long                                    reason-string
   450:21   warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
   456:9    warning  Error message for require is too long                                    reason-string
   457:17   warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
   499:5    warning  Function name must be in mixedCase                                       func-name-mixedcase
   500:5    warning  Function name must be in mixedCase                                       func-name-mixedcase
   517:5    warning  Function name must be in mixedCase                                       func-name-mixedcase
   539:5    warning  Function name must be in mixedCase                                       func-name-mixedcase
   677:1    warning  Contract has 20 states declarations but allowed no more than 15          max-states-count
   708:5    warning  Explicitly mark visibility of state                                      state-visibility
   728:18   warning  Variable name must be in mixedCase                                       var-name-mixedcase
   728:39   warning  Variable name must be in mixedCase                                       var-name-mixedcase
   728:62   warning  Variable name must be in mixedCase                                       var-name-mixedcase
   728:128  warning  Variable name must be in mixedCase                                       var-name-mixedcase
   728:147  warning  Variable name must be in mixedCase                                       var-name-mixedcase
   822:9    warning  Error message for require is too long                                    reason-string
   841:9    warning  Error message for require is too long                                    reason-string
   909:32   warning  Code contains empty blocks                                               no-empty-blocks
   999:9    warning  Error message for require is too long                                    reason-string
  1000:9    warning  Error message for require is too long                                    reason-string
  1011:9    warning  Error message for require is too long                                    reason-string
  1012:9    warning  Error message for require is too long                                    reason-string
  1013:9    warning  Error message for require is too long                                    reason-string
  1015:13   warning  Error message for require is too long                                    reason-string
  1089:13   warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  1104:13   warning  Avoid to make time-based decisions in your business logic                not-rely-on-time

✕ 30 problems (1 error, 29 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 high and 1 Medium and 2 Low and 5 Low issues 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