QuillAudits

Audit Report
January, 2023

For

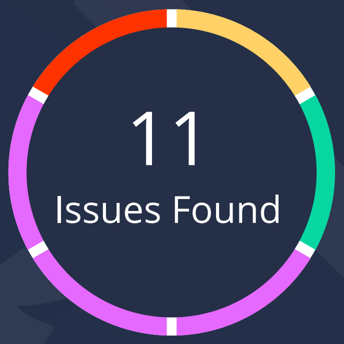
YUG

NEW ERA OF DIGITAL CURRENCIES

Table of Content

Executive Summary

Project Name	YugCoin
Overview	YugCoin is a fork of the SafeMoon project with the extra functionality of minting and burning tokens.
Timeline	October 31, 2022 - December 8, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze YugCoin codebase for quality, security, correctness, and exchange listing. *https://bscscan.com/address/0xF3FD85Ec9eAf17e469Ebad5D6027282ebc7a5eF2#code*
Contracts in Scope	YUG.sol
Fixed In	*https://bscscan.com/address/0x04c8238663631cD88C0a2Dd9f27B6655B6a83496#code*



🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	**1**	0	0	**2**
Partially Resolved Issues	0	0	0	0
Resolved Issues	**1**	**2**	**2**	**3**

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

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 the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

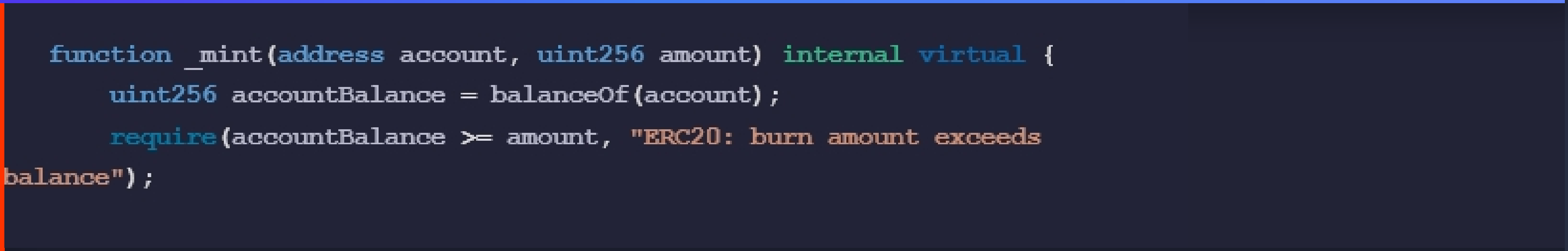
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - YUG.sol

High Severity Issues

1. Infinite mint



```
    function _mint(address account, uint256 amount) internal virtual {
        uint256 accountBalance = balanceOf(account);
        require(accountBalance >= amount, "ERC20: burn amount exceeds
balance");
```

Description

The owner of this contract can mint an infinite number of tokens because there's no cap to the supply or check for how much can be minted. The mint function allows only the owner mint any amount of tokens at any time.
Also, minting requires that the owner holds more tokens than intended to mint. Check the mint function for "accountBalance = accountBalance + amount", it is redundant and would not be used.

Remediation

Place a cap to the amount of tokens that can be minted, or set up a governance system to approve and accept future mints if intended to happen. Review the logic in the mint function as well.

Status

Resolved

2. Centralization issues

Description

All power and core functionality is centralized in the owner. If the owner account gets compromised, the entire project is at risk of token loss, price manipulation, denial of service.

The addLiquidityETH function call sends LP tokens to owner(), if owner is compromised, LP tokens can be mismanaged as well which could lead to price manipulation.

Remediation

Have the private key of the owner account carefully managed, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Have the LP tokens sent to an account with more control, for example: the contract itself with checks to ensure the tokens are not mismanaged.

Status

Acknowledged

Comment: Team will be utilizing Gnosis Safe to lower the risks of a centralization exploit.

Medium Severity Issues

3. Fee limits not coded into contract logic

Description

The whitepaper describes fees per transaction to be 3-5%. It is possible for the owner to set fees to >98% thereby siphoning funds from all transactions.

Remediation

Include a parity check in the setLiquidityFeePercent() and setTaxFeePercent() to match the parameters specified in the whitepaper, so fees would not completely overshadow the transaction value.

Status

Resolved

4. BNB stuck in contract can’t be withdrawn

Description

When the swapAndLiquify function is called, the token swap (YUG to BNB) that occurs results in a slight price change that makes it require less BNB to stake the other half of the YUG balance. Thus leaving a small amount of BNB left in the contract with no function to get those BNB tokens out. Although none of the functions in the scope of this contract are dependent on the contract BNB balance, the tokens would be technically unusable.

Remediation

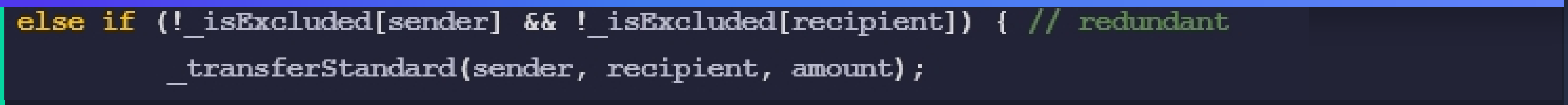
Make a function to clear out the stuck BNB tokens or find some way to fit into your proposed logic (e.g. distribute to YUG holders, use to buyback YUG tokens etc)

Status

Resolved

Low Severity Issues

5. Redundant check in tokenTransfer function



```
else if (!_isExcluded[sender] && !_isExcluded[recipient]) { // redundant
            _transferStandard(sender, recipient, amount);
```

Description

With all other checks done, !isExcluded[sender] && !isExcluded[recipient] performs the same function as the else clause.

Recommendation

Consider removing the redundant check here since it is covered in the else clause.

Status

Resolved

6. Unused code

Description

The deliver() function is defined but never implemented in the scope of the codebase. It can be called by any user, but its use is not entirely clear.

Remediation

Consider removing if not needed.

Status

Resolved

Informational Issues

7. Inaccurate error message

```
function includeInReward(address account) public onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
```

```
function _mint(address account, uint256 amount) internal virtual {
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
```

Description

The includeInReward function checks if the account address passed in is not already excluded. If (_isExcluded[account]) returns true, the program would run because the address checked is not excluded. If (_isExcluded[account]) returns false, the program should give an error that says “Account is not excluded” instead).

Same with the error message in the _mint function.

Recommendation

Update the error message to give a more accurate response.

Status

Resolved

8. Unlocked pragma (pragma solidity ^0.6.12)

Description

The solidity pragma version in this codebase is unlocked.

Remediation

It is always advisable to use a specific version when deploying to production to reduce the surface of attack with future releases.

Status

Resolved

9. Static variables can be set to constant

```
string constant _name = "YUG";
string constant _symbol = "YUG";
uint8 constant _decimals = 18;
```

Description

The private variables (name, symbol and decimals) can save a little more gas at their declaration. If the total supply as well would be made static, it should be made constant too.

Recommendation

Consider making the variables constant.

Status

Resolved

10. Unhandled return values

Description

The return values from the addLiquidityETH function call are not properly handled.

Remediation

If the success or failure of the function call should cause differences in response consider using variables to handle either success or failure.

Status

Acknowledged

11. Naming convention

Description

The codebase is built around Binance Smart Chain not Ethereum. Although there's no difference in functionality, the syntax of the code contains "Uniswap" and "ETH" instead of "PancakeSwap" and BNB.

Also, the setMaxTxPercent() function sets the maxTxAmount variable but doesn't adjust transaction limits.

Recommendation

Consider renaming the functions and variables to match the deployment environment.

Also have the setMaxTxPercent() function named according to what it should do.

Status

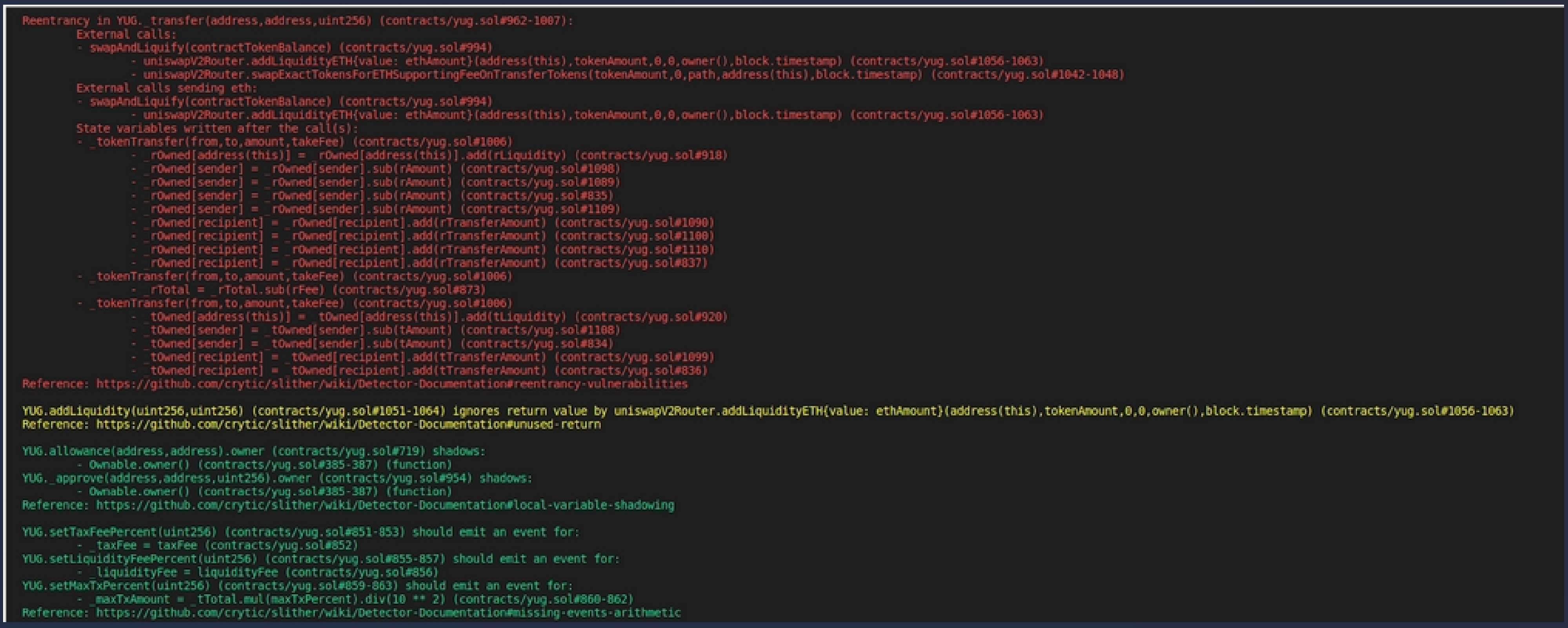
Acknowledged

Functional Testing

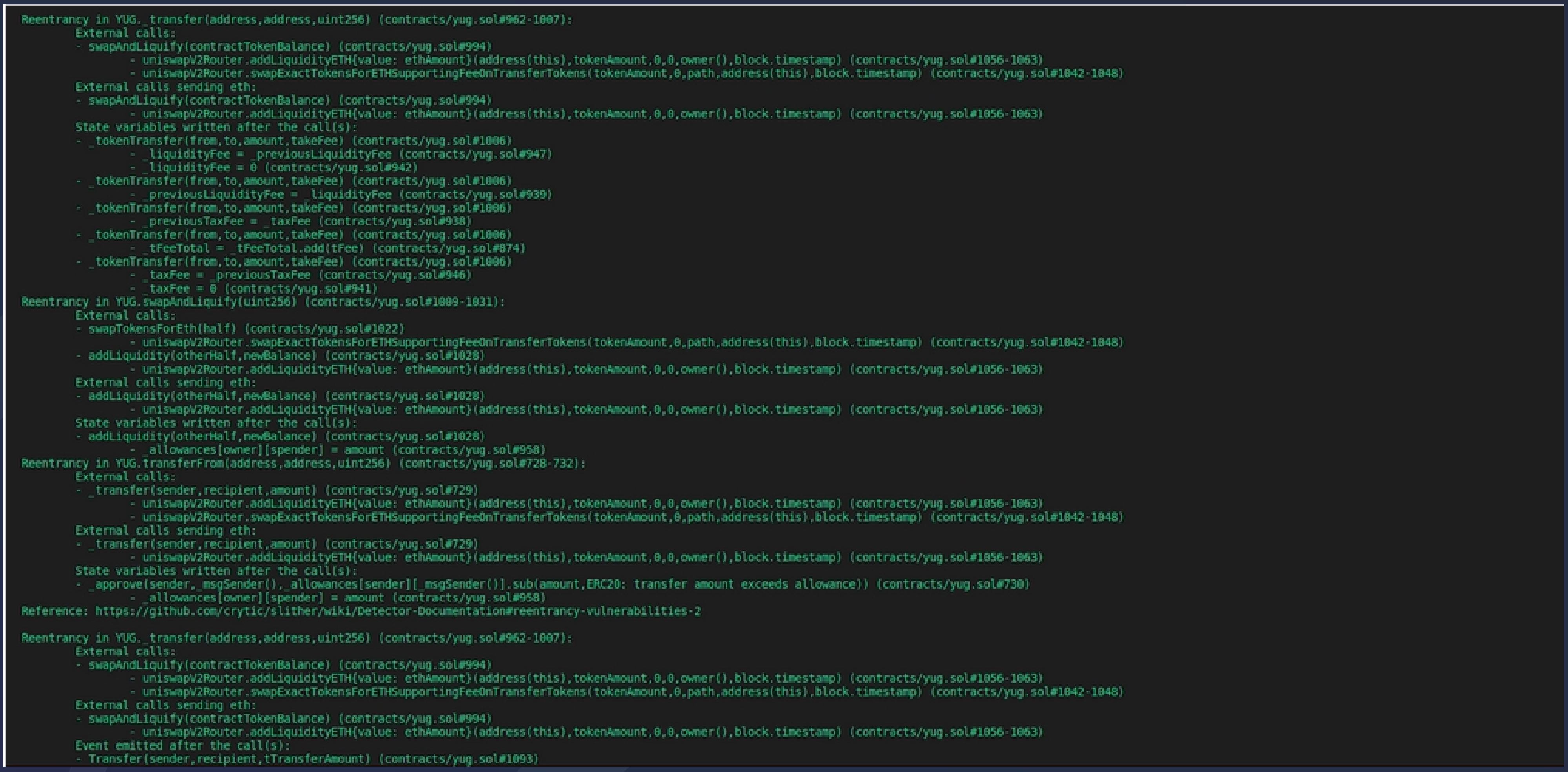
Some of the tests performed are mentioned below:

- ✓ should return token name
- ✓ should not exceed total supply
- ✗ should be able to renounce ownership
- ✗ should only allow max tx. tax within 5% as stated in whitepaper

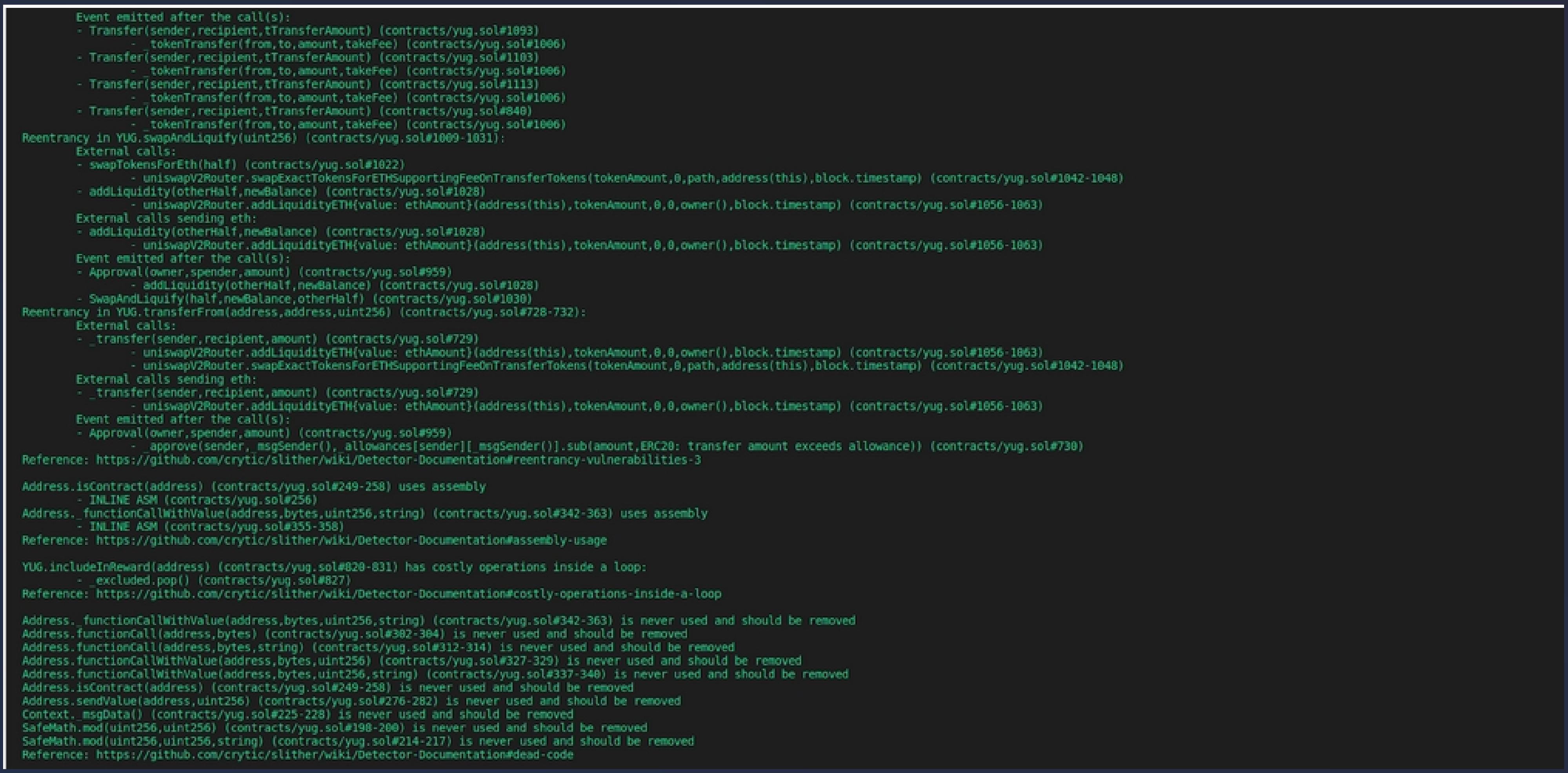
Automated Tests



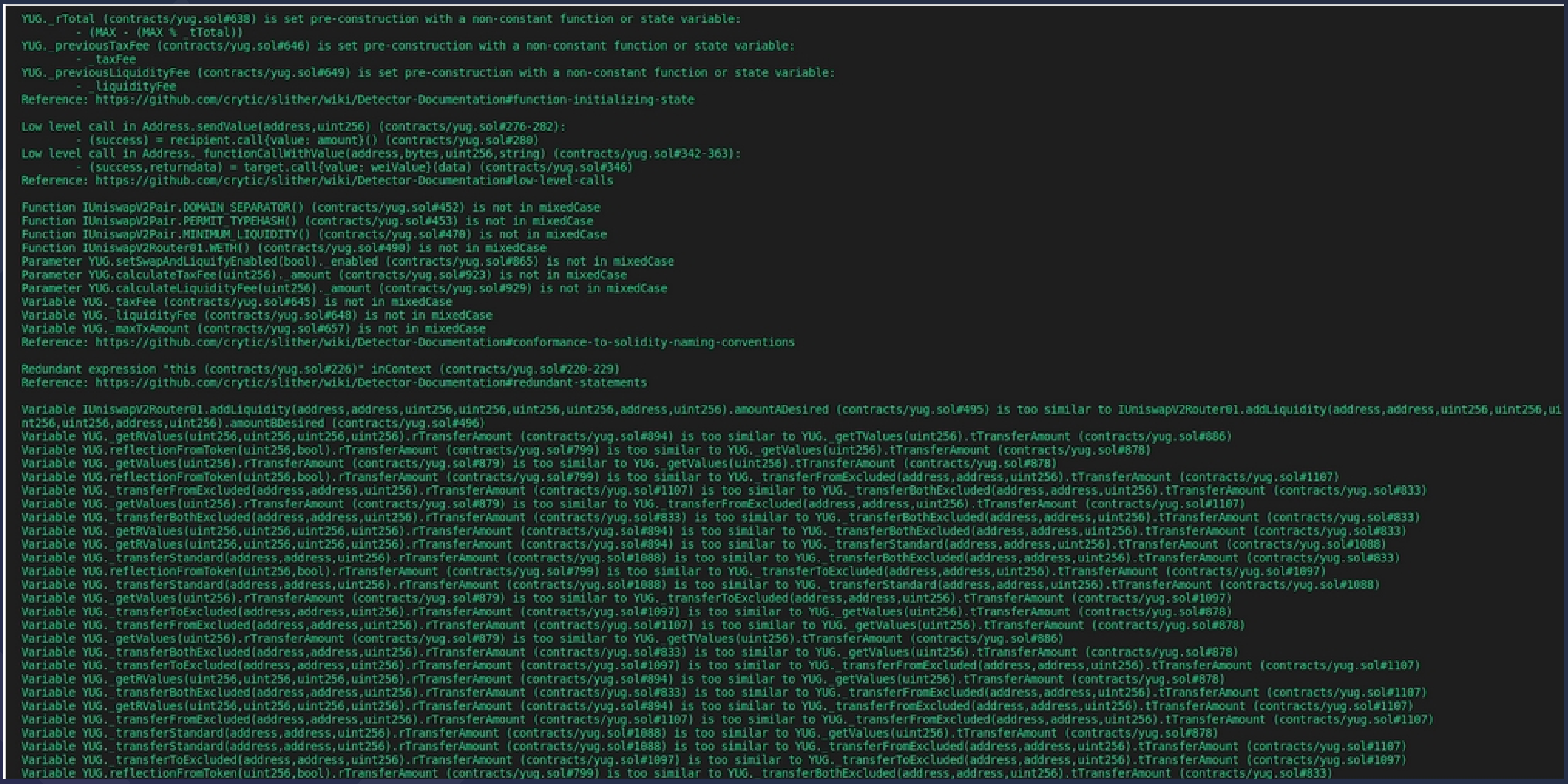
```
Reentrancy in YUG._transfer(address,address,uint256) (contracts/yug.sol#962-1007):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (contracts/yug.sol#918)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/yug.sol#1098)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/yug.sol#1089)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/yug.sol#835)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/yug.sol#1109)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/yug.sol#1090)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/yug.sol#1100)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/yug.sol#1110)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/yug.sol#837)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _rTotal = _rTotal.sub(rFee) (contracts/yug.sol#873)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (contracts/yug.sol#920)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/yug.sol#1108)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/yug.sol#834)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (contracts/yug.sol#1099)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (contracts/yug.sol#836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

YUG.addLiquidity(uint256,uint256) (contracts/yug.sol#1051-1064) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

YUG.allowance(address,address).owner (contracts/yug.sol#719) shadows:
        - Ownable.owner() (contracts/yug.sol#385-387) (function)
YUG._approve(address,address,uint256).owner (contracts/yug.sol#954) shadows:
        - Ownable.owner() (contracts/yug.sol#385-387) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

YUG.setTaxFeePercent(uint256) (contracts/yug.sol#851-853) should emit an event for:
        - _taxFee = taxFee (contracts/yug.sol#852)
YUG.setLiquidityFeePercent(uint256) (contracts/yug.sol#855-857) should emit an event for:
        - _liquidityFee = liquidityFee (contracts/yug.sol#856)
YUG.setMaxTxPercent(uint256) (contracts/yug.sol#859-863) should emit an event for:
        - _maxTxAmount = _tTotal.mul(maxTxPercent).div(10 ** 2) (contracts/yug.sol#860-862)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```



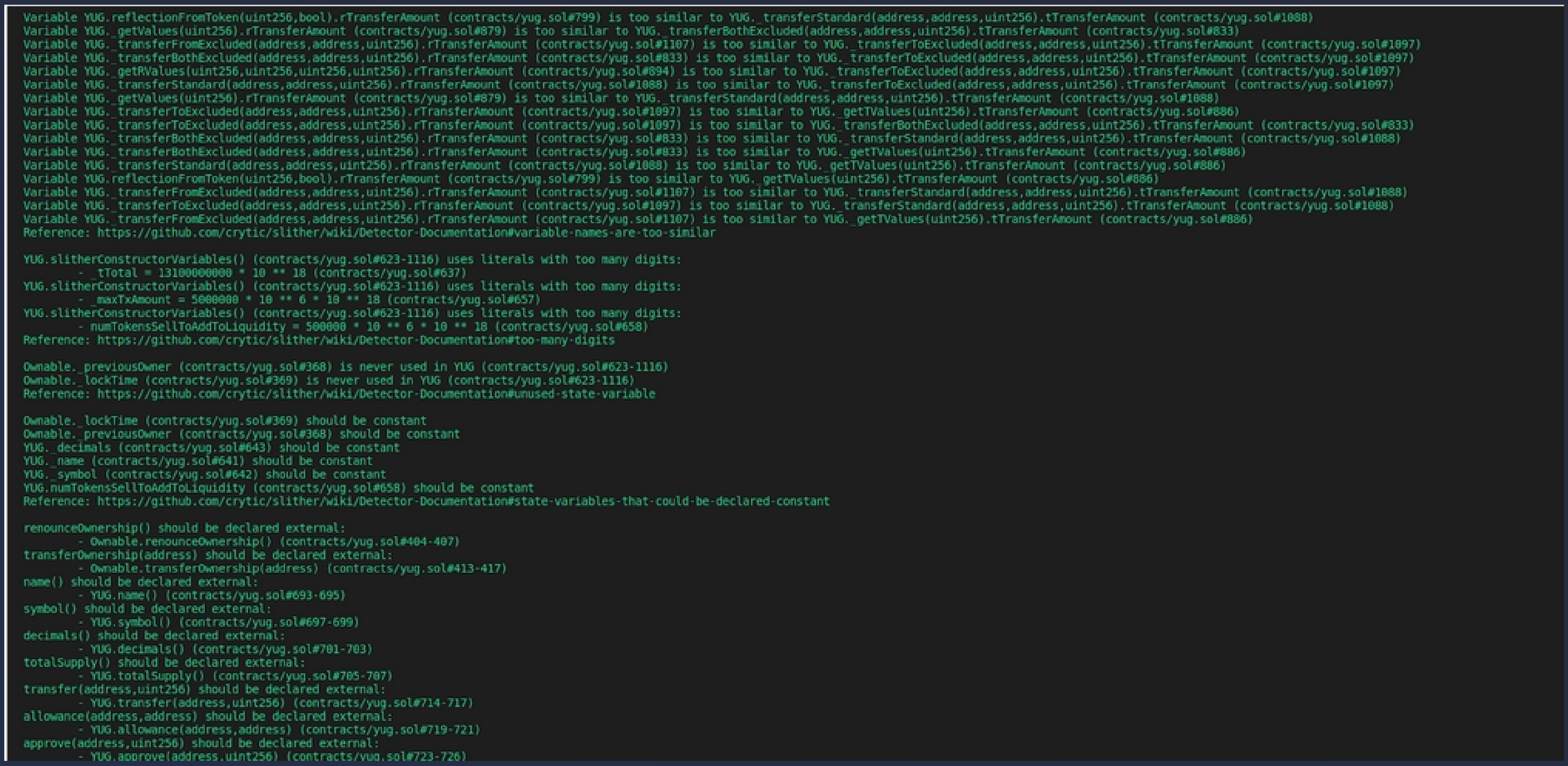
```
Reentrancy in YUG._transfer(address,address,uint256) (contracts/yug.sol#962-1007):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _liquidityFee = _previousLiquidityFee (contracts/yug.sol#947)
                - _liquidityFee = 0 (contracts/yug.sol#942)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _previousLiquidityFee = _liquidityFee (contracts/yug.sol#939)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _previousTaxFee = _taxFee (contracts/yug.sol#938)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _tFeeTotal = _tFeeTotal.add(tFee) (contracts/yug.sol#874)
        - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
                - _taxFee = _previousTaxFee (contracts/yug.sol#946)
                - _taxFee = 0 (contracts/yug.sol#941)
Reentrancy in YUG.swapAndLiquify(uint256) (contracts/yug.sol#1009-1031):
        External calls:
        - swapTokensForEth(half) (contracts/yug.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
                - _allowances[owner][spender] = amount (contracts/yug.sol#958)
Reentrancy in YUG.transferFrom(address,address,uint256) (contracts/yug.sol#728-732):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/yug.sol#729)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/yug.sol#729)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/yug.sol#730)
                - _allowances[owner][spender] = amount (contracts/yug.sol#958)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in YUG._transfer(address,address,uint256) (contracts/yug.sol#962-1007):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (contracts/yug.sol#994)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (contracts/yug.sol#1093)
```

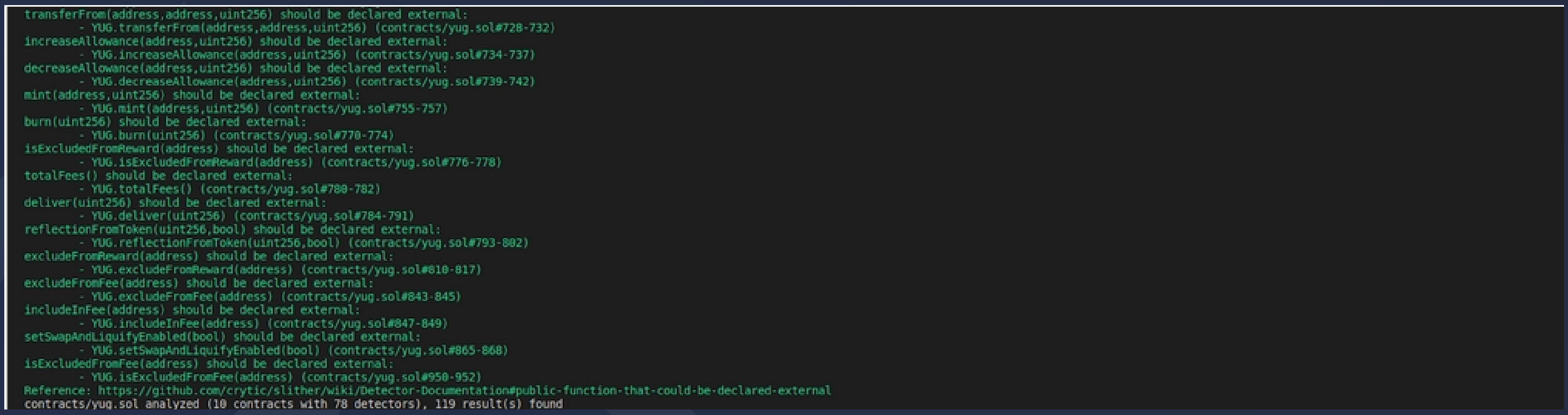


```
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (contracts/yug.sol#1093)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
        - Transfer(sender,recipient,tTransferAmount) (contracts/yug.sol#1103)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
        - Transfer(sender,recipient,tTransferAmount) (contracts/yug.sol#1113)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
        - Transfer(sender,recipient,tTransferAmount) (contracts/yug.sol#840)
                - _tokenTransfer(from,to,amount,takeFee) (contracts/yug.sol#1006)
Reentrancy in YUG.swapAndLiquify(uint256) (contracts/yug.sol#1009-1031):
        External calls:
        - swapTokensForEth(half) (contracts/yug.sol#1022)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/yug.sol#959)
                - addLiquidity(otherHalf,newBalance) (contracts/yug.sol#1028)
        - SwapAndLiquify(half,newBalance,otherHalf) (contracts/yug.sol#1030)
Reentrancy in YUG.transferFrom(address,address,uint256) (contracts/yug.sol#728-732):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/yug.sol#729)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/yug.sol#1042-1048)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/yug.sol#729)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/yug.sol#1056-1063)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/yug.sol#959)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/yug.sol#730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.isContract(address) (contracts/yug.sol#249-258) uses assembly
        - INLINE ASM (contracts/yug.sol#256)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/yug.sol#342-363) uses assembly
        - INLINE ASM (contracts/yug.sol#355-358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

YUG.includeInReward(address) (contracts/yug.sol#820-831) has costly operations inside a loop:
        - _excluded.pop() (contracts/yug.sol#827)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Address._functionCallWithValue(address,bytes,uint256,string) (contracts/yug.sol#342-363) is never used and should be removed
Address.functionCall(address,bytes) (contracts/yug.sol#302-304) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/yug.sol#312-314) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/yug.sol#327-329) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/yug.sol#337-340) is never used and should be removed
Address.isContract(address) (contracts/yug.sol#249-258) is never used and should be removed
Address.sendValue(address,uint256) (contracts/yug.sol#276-282) is never used and should be removed
Context._msgData() (contracts/yug.sol#225-228) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/yug.sol#198-200) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/yug.sol#214-217) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



```
YUG._rTotal (contracts/yug.sol#638) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
YUG._previousTaxFee (contracts/yug.sol#646) is set pre-construction with a non-constant function or state variable:
        - _taxFee
YUG._previousLiquidityFee (contracts/yug.sol#649) 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

Low level call in Address.sendValue(address,uint256) (contracts/yug.sol#276-282):
        - (success) = recipient.call{value: amount}() (contracts/yug.sol#280)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (contracts/yug.sol#342-363):
        - (success,returndata) = target.call{value: weiValue}(data) (contracts/yug.sol#346)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/yug.sol#452) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/yug.sol#453) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/yug.sol#470) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/yug.sol#490) is not in mixedCase
Parameter YUG.setSwapAndLiquifyEnabled(bool)._enabled (contracts/yug.sol#865) is not in mixedCase
Parameter YUG.calculateTaxFee(uint256)._amount (contracts/yug.sol#923) is not in mixedCase
Parameter YUG.calculateLiquidityFee(uint256)._amount (contracts/yug.sol#929) is not in mixedCase
Variable YUG._taxFee (contracts/yug.sol#645) is not in mixedCase
Variable YUG._liquidityFee (contracts/yug.sol#648) is not in mixedCase
Variable YUG._maxTxAmount (contracts/yug.sol#657) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/yug.sol#226)" inContext (contracts/yug.sol#220-229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/yug.sol#495) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,ui
nt256,uint256,address,uint256).amountBDesired (contracts/yug.sol#496)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._getValues(uint256).tTransferAmount (contracts/yug.sol#878)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1107)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
```


```
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/yug.sol#894) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._transferToExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#1097)
Variable YUG._getValues(uint256).rTransferAmount (contracts/yug.sol#879) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/yug.sol#833)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#833) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG._transferStandard(address,address,uint256).rTransferAmount (contracts/yug.sol#1088) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG.reflectionFromToken(uint256,bool).rTransferAmount (contracts/yug.sol#799) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._transferToExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1097) is too similar to YUG._transferStandard(address,address,uint256).tTransferAmount (contracts/yug.sol#1088)
Variable YUG._transferFromExcluded(address,address,uint256).rTransferAmount (contracts/yug.sol#1107) is too similar to YUG._getTValues(uint256).tTransferAmount (contracts/yug.sol#886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

YUG.slitherConstructorVariables() (contracts/yug.sol#623-1116) uses literals with too many digits:
        - _tTotal = 13100000000 * 10 ** 18 (contracts/yug.sol#637)
YUG.slitherConstructorVariables() (contracts/yug.sol#623-1116) uses literals with too many digits:
        - _maxTxAmount = 5000000 * 10 ** 6 * 10 ** 18 (contracts/yug.sol#657)
YUG.slitherConstructorVariables() (contracts/yug.sol#623-1116) uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 500000 * 10 ** 6 * 10 ** 18 (contracts/yug.sol#658)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Ownable._previousOwner (contracts/yug.sol#368) is never used in YUG (contracts/yug.sol#623-1116)
Ownable._lockTime (contracts/yug.sol#369) is never used in YUG (contracts/yug.sol#623-1116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Ownable._lockTime (contracts/yug.sol#369) should be constant
Ownable._previousOwner (contracts/yug.sol#368) should be constant
YUG._decimals (contracts/yug.sol#643) should be constant
YUG._name (contracts/yug.sol#641) should be constant
YUG._symbol (contracts/yug.sol#642) should be constant
YUG.numTokensSellToAddToLiquidity (contracts/yug.sol#658) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/yug.sol#404-407)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/yug.sol#413-417)
name() should be declared external:
        - YUG.name() (contracts/yug.sol#693-695)
symbol() should be declared external:
        - YUG.symbol() (contracts/yug.sol#697-699)
decimals() should be declared external:
        - YUG.decimals() (contracts/yug.sol#701-703)
totalSupply() should be declared external:
        - YUG.totalSupply() (contracts/yug.sol#705-707)
transfer(address,uint256) should be declared external:
        - YUG.transfer(address,uint256) (contracts/yug.sol#714-717)
allowance(address,address) should be declared external:
        - YUG.allowance(address,address) (contracts/yug.sol#719-721)
approve(address,uint256) should be declared external:
        - YUG.approve(address,uint256) (contracts/yug.sol#723-726)
```


```
transferFrom(address,address,uint256) should be declared external:
        - YUG.transferFrom(address,address,uint256) (contracts/yug.sol#728-732)
increaseAllowance(address,uint256) should be declared external:
        - YUG.increaseAllowance(address,uint256) (contracts/yug.sol#734-737)
decreaseAllowance(address,uint256) should be declared external:
        - YUG.decreaseAllowance(address,uint256) (contracts/yug.sol#739-742)
mint(address,uint256) should be declared external:
        - YUG.mint(address,uint256) (contracts/yug.sol#755-757)
burn(uint256) should be declared external:
        - YUG.burn(uint256) (contracts/yug.sol#770-774)
isExcludedFromReward(address) should be declared external:
        - YUG.isExcludedFromReward(address) (contracts/yug.sol#776-778)
totalFees() should be declared external:
        - YUG.totalFees() (contracts/yug.sol#780-782)
deliver(uint256) should be declared external:
        - YUG.deliver(uint256) (contracts/yug.sol#784-791)
reflectionFromToken(uint256,bool) should be declared external:
        - YUG.reflectionFromToken(uint256,bool) (contracts/yug.sol#793-802)
excludeFromReward(address) should be declared external:
        - YUG.excludeFromReward(address) (contracts/yug.sol#810-817)
excludeFromFee(address) should be declared external:
        - YUG.excludeFromFee(address) (contracts/yug.sol#843-845)
includeInFee(address) should be declared external:
        - YUG.includeInFee(address) (contracts/yug.sol#847-849)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - YUG.setSwapAndLiquifyEnabled(bool) (contracts/yug.sol#865-868)
isExcludedFromFee(address) should be declared external:
        - YUG.isExcludedFromFee(address) (contracts/yug.sol#950-952)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
contracts/yug.sol analyzed (10 contracts with 78 detectors), 119 result(s) found
```

Closing Summary

In this report, we have considered the security of YugCoin. We performed our audit according to the procedure described above.

Some Issues of High, Medium, Low and informational severity were found during the course of the audit.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the YugCoin Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the YugCoin Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
January, 2023

For

YUG
NEW ERA OF DIGITAL CURRENCIES

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com