QuillAudits

Audit Report
January, 2022

For

ACYC

Contents

Scope of the Audit

The scope of this audit was to analyze and document the ACYC Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

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, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
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.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	3	2	2
Closed	0	1	0	0

Introduction

During the period of **Dec 16, 2021 to Jan 25, 2022** - QuillAudits Team performed a security audit for ACYC token smart contracts.

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

V	Date	Contract address
1	Dec 20	https://etherscan.io/address/0xb56a1f3310578f23120182fb2e58c087efe6e147#code

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

1. Missing Check for Reentrancy Attack

Calling acyc._transfer() might trigger function uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens() , which is implemented by 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 being lost or stolen.

Remediation
We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the acyc._transfer() function to prevent reentrancy attacks. Moreover, the Auditee should also review the address and assess the risks of third-party dependencies carefully before deployment.

Status: Acknowledged
The auditee confirmed that the issue shall be fixed in the next version.

2. Centalization risk

The role owner has the authority to update these critical settings:
- setTreasuryAddress()
- excludeFromTaxes()
- setReflectionsTax()
- setTreasuryTax()
- manualSwap()
- manualSend()

Remediation
We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

- with reasonable latency for community awareness on privileged operations;
- Multisig with community-voted 3rd-party independent co-signers;
- DAO or Governance module increasing transparency and community involvement;

Status: Acknowledged
The auditee confirmed that the issue shall be fixed in the next version.

3. Lack of event emissions

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- setTreasuryAddress()
- excludeFromTaxes()
- setReflectionsTax()
- setTreasuryTax()

Remediation
We recommend emitting an event to log the update of the variables for the aforementioned functions.

Status: Acknowledged
The auditee confirmed that the issue shall be fixed in the next version.

4. transfer() is being utilized

Low-level transfer() function has been found to be used in the contract at line 504:

_treasuryAddress.transfer(amount);

Due to the fact that .transfer() and .send() forward exactly 2,300 gas to the recipient. This hardcoded gas stipend aimed to prevent reentrancy vulnerabilities, but this only makes sense under the assumption that gas costs are constant. Recently EIP 1884 was included in the Istanbul hard fork. One of the changes included in EIP 1884 is an increase to the gas cost of the SLOAD operation, causing a contract's fallback function to cost more than 2300 gas.

```
// bad
contract Vulnerable {
    function withdraw(uint256 amount) external {
        // This forwards 2300 gas, which may not be enough if the recipient
        // is a contract and gas costs change.
        msg.sender.transfer(amount);
    }
}

// good
contract Fixed {
    function withdraw(uint256 amount) external {
        // This forwards all available gas. Be sure to check the return value!
        (bool success, ) = msg.sender.call{value:amount}("");
        require(success, "Transfer failed.");
    }
}
```

Remediation
The auditee needs to ensure that the _treasuryAddress is not a contract .On the other hand, it's recommended to stop using .transfer() and .send() and instead use .call().

Status: Fixed
The auditee ensured that the _treasuryAddress is not a contract

Low severity issues

5. Uninformative revert messages in require statements

Description
The error message in require(_msgSender() == _treasuryAddress, "You cannot call this"); does not describe the error correctly.
Uninformative error messages greatly damage the overall user experience, thus lowering the system's quality.

Remediation
We recommend changing "You cannot call this" to "You're not the treasury address" . Consider not only fixing the specific instance mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Acknowledged

6. Tautology or Contradiction Issue

Line	Code
152	require(bool,string)(tax >= 0 && tax <= 10,ERC20: tax out of band)
158	require(bool,string)(tax >= 0 && tax <= 10,ERC20: tax out of band)

Description
Tax is a uint256, so tax >= 0 will be always true. This comparison is a tautology that will waste gas during the execution.

Remediation
Fix the incorrect comparison by changing the value type or the comparison.

Status: Acknowledged
The auditee confirmed that the issue shall be fixed in the next version.

Informational issues

7. Not using delete to zero values

In the _setNoFees function within the contract, the following variables are manually replaced with Zero:
_currentTaxForReflections = 0;
_currentTaxForTreasury = 0;

Similar to the _setSellSideFees() and _setBuySideFees() function, _currentTaxForReflections and _currentTaxForTreasury are respectively replaced with zero.

To simplify the code and clarify intent, consider using delete instead.

Status: Acknowledged

8. Typos

Please consider changing the following typos:

- L124: recieve should be received
- L124: swaping should be swapping
- L446: transfered should be transferred
- L449: totaly should be totally
- L521: transfering should be transferring
- L574: necssary should be necessary

Status: Acknowledged
The auditee confirmed that the issue shall be fixed in the next version.

Functional Tests

Evn: Ropsten
Contract: 0xCc96c380cd64074A59A373882ce948474a6Ffc64

Function name	Input	Output	TX	Status
setTreasuryAddress (only called by _treasuryAddress)	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	True	0xf7cf7e0446524777ccb94c89ae8a0a5150a886846a0ff1bd8911d6fa746aee69	Passed
excludeFromTaxes	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC, false	true	0xdae4e8b070e92e694d7ac31f23762301d0bc7fb35e0e908e9fe6f700567d8cb6	Passed
setReflectionsTax	8	true	0x5867868fb740576cf1f210052e19908939ac7749dd73b5673e6315b12693db5b	Passed
setTreasuryTax	8	true	0x847f98921a50c8ffa0b3d0967cc552165893bc20a7cfc66bd624d06577b6298f	Passed
getETHBalance	500000000000000000	true	N/A	Passed
manualSend	N/A	True	0x59ba9366ce2de35ba8571c6e84a93f4752b0a2d7b1a3e270b48accdfe37bbbba	Passed
getETHBalance	0	True	N/A	Passed
transfer	"0xCc96c380cd64074A59A373882ce948474a6Ffc64","500000000000000000"	true	0xf74e81072c3a376313bc704278c39b01149a25234b21814df7ef261c64765040	Passed

balanceOf	0xCc96c380cd64074A59A373882ce948474a6Ffc64	5000000000000000	N/A	Passed
approve	"0x153b057d5d7262dC92099B59c975255ecE66784F","100000000000000000000000000"	true	0x63d60667f30959820c03c87b8752af59c40cdfee14245b7493a6296c6e05cd08	Passed
allowance	"0x1dd64394E29c5988f04A8E074D0DBACd4D614729","0x153b057d5d7262dC92099B59c975255ecE66784F"	10000000000000000000000000000	N/A	Passed
decreaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","100000000000000000000000000"	true	0xa0cebd892cc62c184c00de12b5293a280be9d829e5db20fa057e0b7268a83a72	Passed
allowance	"0x1dd64394E29c5988f04A8E074D0DBACd4D614729","0x153b057d5d7262dC92099B59c975255ecE66784F"	0	N/A	Passed
increaseAllowance	"0x153b057d5d7262dC92099B59c975255ecE66784F","100000000000000000000000000"	true	0xc1b3d294911098ebe97ee04b2c43396aaacea1ceedf021e6cb6b645d1d319097	Passed
allowance	"0x1dd64394E29c5988f04A8E074D0DBACd4D614729","0x153b057d5d7262dC92099B59c975255ecE66784F"	10000000000000000000000000000	N/A	Passed
transferFrom	"0x1dd64394E29c5988f04A8E074D0DBACd4D614729","0x153b057d5d7262dC92099B59c975255ecE66784F",5000000000000000000000000	True	0x387e0f7bcaebc300adb88250751904162616d838a3e57c72d9db81a6ae80b04a	Passed
allowance	"0x1dd64394E29c5988f04A8E074D0DBACd4D614729","0x153b057d5d7262dC92099B59c975255ecE66784F"	5000000000000000000000000000	N/A	Passed
transferOwnership	0x60b6D91cB698F41E1eD928f9631cEC6b8Ff8F6cC	true	0xb359cf2f51edb64ab66c04eca989e70	Passed

			19d5828a8aacaf4b261bd132d67c6feea	

Contract: 0x3f398a937AEE1e97dEe9fD197B208A9099F8099C

Function name	Input	Output	TX	Status
transfer	"0x153b057d5d7262dC92099B59c975255ecE66784F","50000000000000000000"	true	0x601ca96c6eeabb4e8126f86a60b02e5baa548396b4b248ffc8525eb4d231ea1d	Passed
transfer	"0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05","10000000000000000000"	true	0xc270085486c2390c73cc95f64eae4fd088b9affa68b82b5dca89539c8f09a472	Passed
balanceOf	0x8cF95C8750FB8E83Cc45F44232da9Dd022037e05	900000000000000000	N/A	Passed
balanceOf (contract)	0x3f398a937AEE1e97dEe9fD197B208A9099F8099C	100000000000000000	N/A	Passed

Contract: 0x176A3d6FCd276255d86D5300C5BDaE5071Ee57f5
addLiquidityETH TX: 0x539b388412bc7c0872a83940724e48af1584a05c8c1ce4e62fc294ce36eb232d
Pool: 0xF614cD8A349458f82310fb4e282964e67Cb57dEa

Function name	Input	Output	TX	Status
manualSwap	N/A	True	0x89326f3417c087f23b00e1be1a0d75002cd66344ea6cca90fef2668fc2f5ba40	Passed
buy	44.422111548087034307	true	0xe3bb80e2da624e1c7861af6abdb5cb43353956ede1265094cabf3b782f9473a1	Passed
sell	44.422111548087034307	true	0x8b11baf16ba90798ba35431768a22c35e5a11422dc130231be290a9865baeca3	Passed

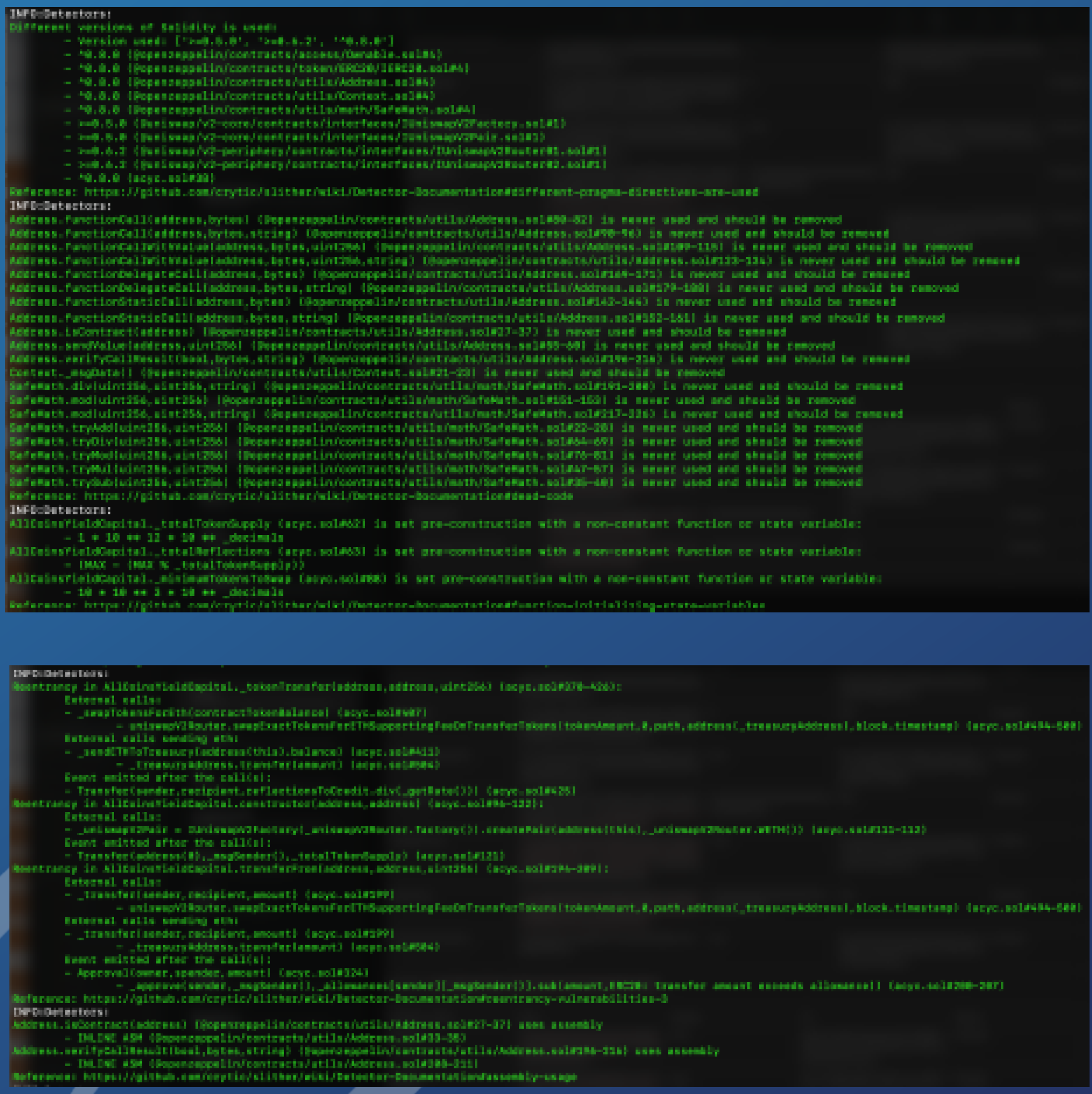
Automated Tests

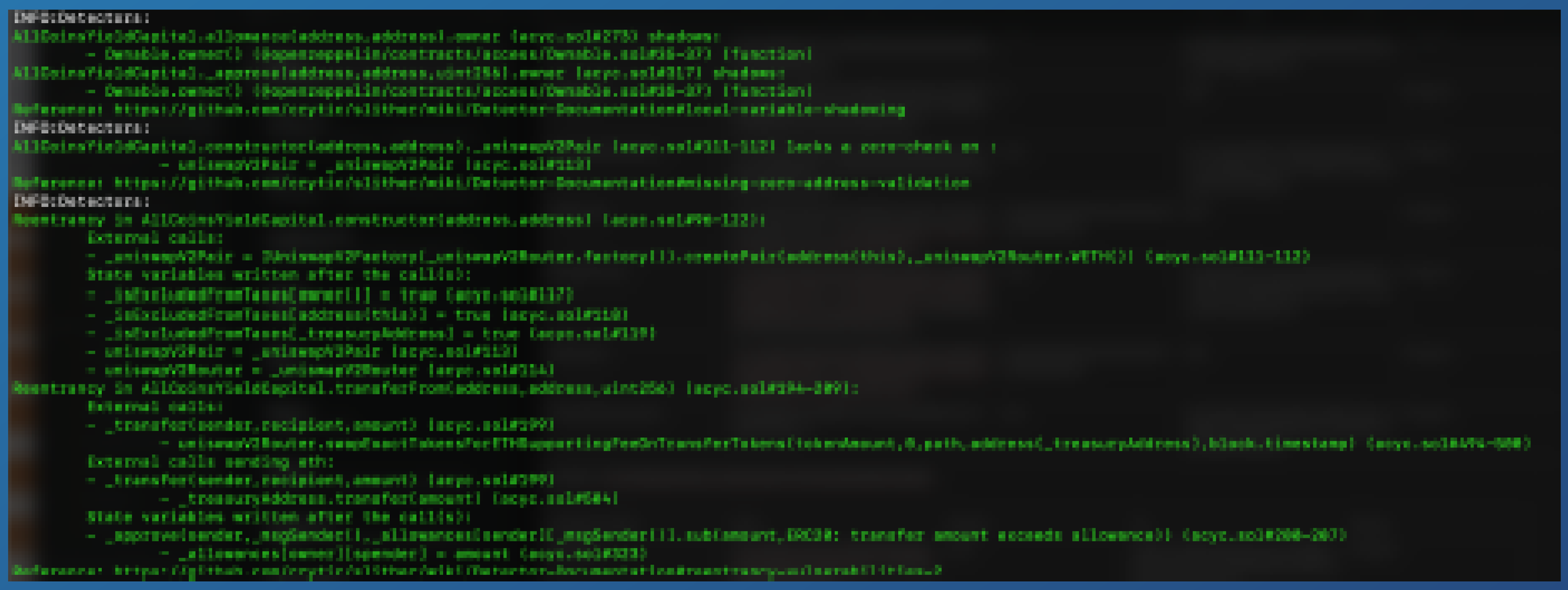
Slither

```
INFO:Detectors:
AllCoinsYieldCapital._sendETHToTreasury(uint256) (acyc.sol#503-505) sends eth to arbitrary user
        Dangerous calls:
        - _treasuryAddress.transfer(amount) (acyc.sol#504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in AllCoinsYieldCapital._tokenTransfer(address,address,uint256) (acyc.sol#370-426):
        External calls:
        - _swapTokensForEth(contractTokenBalance) (acyc.sol#407)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(_treasuryAddress),block.timestamp) (acyc.sol#494-500)
        External calls sending eth:
        - _sendETHToTreasury(address(this).balance) (acyc.sol#411)
                - _treasuryAddress.transfer(amount) (acyc.sol#504)
        State variables written after the call(s):
        - _reflectionsOwned[sender] = _reflectionsOwned[sender].sub(reflectionsToDebit) (acyc.sol#417-419)
        - _reflectionsOwned[recipient] = _reflectionsOwned[recipient].add(reflectionsToCredit) (acyc.sol#420-422)
Reentrancy in AllCoinsYieldCapital._transfer(address,address,uint256) (acyc.sol#330-367):
        External calls:
        - _tokenTransfer(sender,recipient,amountOfTokens) (acyc.sol#363)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(_treasuryAddress),block.timestamp) (acyc.sol#494-500)
        External calls sending eth:
        - _tokenTransfer(sender,recipient,amountOfTokens) (acyc.sol#363)
                - _treasuryAddress.transfer(amount) (acyc.sol#504)
        State variables written after the call(s):
        - _restoreAllFees() (acyc.sol#366)
                - _currentTaxForReflections = _fixedTaxForReflections (acyc.sol#533)
        - _restoreAllFees() (acyc.sol#366)
                - _currentTaxForTreasury = _fixedTaxForTreasury (acyc.sol#534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
AllCoinsYieldCapital.setReflectionsTax(uint256) (acyc.sol#151-155) contains a tautology or contradiction:
        - require(bool,string)(tax >= 0 && tax <= 10,ERC20: tax out of band) (acyc.sol#152)
AllCoinsYieldCapital.setTreasuryTax(uint256) (acyc.sol#157-161) contains a tautology or contradiction:
        - require(bool,string)(tax >= 0 && tax <= 10,ERC20: tax out of band) (acyc.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#62-65)
transfer(address,uint256) should be declared external:
        - AllCoinsYieldCapital.transfer(address,uint256) (acyc.sol#175-182)
approve(address,uint256) should be declared external:
        - AllCoinsYieldCapital.approve(address,uint256) (acyc.sol#184-191)
transferFrom(address,address,uint256) should be declared external:
        - AllCoinsYieldCapital.transferFrom(address,address,uint256) (acyc.sol#194-209)
increaseAllowance(address,uint256) should be declared external:
        - AllCoinsYieldCapital.increaseAllowance(address,uint256) (acyc.sol#211-222)
decreaseAllowance(address,uint256) should be declared external:
        - AllCoinsYieldCapital.decreaseAllowance(address,uint256) (acyc.sol#224-238)
name() should be declared external:
        - AllCoinsYieldCapital.name() (acyc.sol#240-242)
symbol() should be declared external:
        - AllCoinsYieldCapital.symbol() (acyc.sol#244-246)
decimals() should be declared external:
        - AllCoinsYieldCapital.decimals() (acyc.sol#248-250)
totalSupply() should be declared external:
        - AllCoinsYieldCapital.totalSupply() (acyc.sol#252-255)
isExcludedFromTaxes(address) should be declared external:
        - AllCoinsYieldCapital.isExcludedFromTaxes(address) (acyc.sol#257-259)
totalTaxesSentToReflections() should be declared external:
        - AllCoinsYieldCapital.totalTaxesSentToReflections() (acyc.sol#261-263)
totalTaxesSentToTreasury() should be declared external:
        - AllCoinsYieldCapital.totalTaxesSentToTreasury() (acyc.sol#265-267)
getETHBalance() should be declared external:
        - AllCoinsYieldCapital.getETHBalance() (acyc.sol#269-271)
allowance(address,address) should be declared external:
        - AllCoinsYieldCapital.allowance(address,address) (acyc.sol#273-280)
reflectionFromToken(uint256,bool) should be declared external:
        - AllCoinsYieldCapital.reflectionFromToken(uint256,bool) (acyc.sol#286-301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

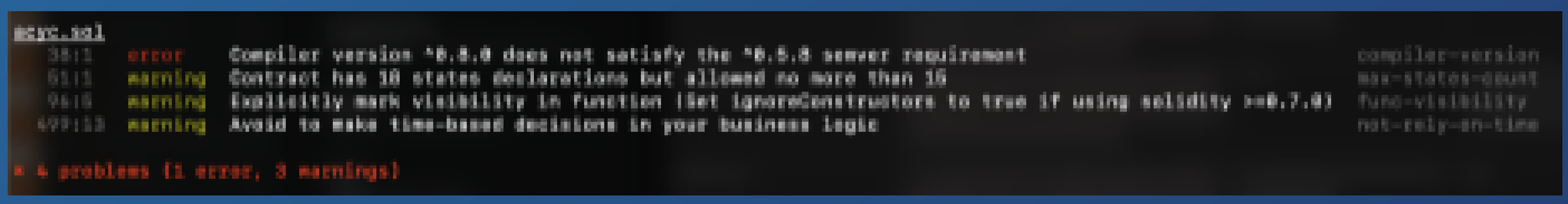
```
INFO:Detectors:
Reentrancy in AllCoinsYieldCapital._tokenTransfer(address,address,uint256) (acyc.sol#370-426):
        External calls:
        - _sendETHToTreasury(address(this).balance) (acyc.sol#411)
                - _treasuryAddress.transfer(amount) (acyc.sol#504)
        State variables written after the call(s):
        - _reflectionsOwned[sender] = _reflectionsOwned[sender].sub(reflectionsToDebit) (acyc.sol#417-419)
        - _reflectionsOwned[recipient] = _reflectionsOwned[recipient].add(reflectionsToCredit) (acyc.sol#420-422)
        Event emitted after the call(s):
        - Transfer(sender,recipient,reflectionsToCredit.div(_getRate())) (acyc.sol#425)
Reentrancy in AllCoinsYieldCapital._transfer(address,address,uint256) (acyc.sol#330-367):
        External calls:
        - _tokenTransfer(sender,recipient,amountOfTokens) (acyc.sol#363)
                - _treasuryAddress.transfer(amount) (acyc.sol#504)
        State variables written after the call(s):
        - _restoreAllFees() (acyc.sol#366)
                - _currentTaxForReflections = _fixedTaxForReflections (acyc.sol#533)
        - _restoreAllFees() (acyc.sol#366)
                - _currentTaxForTreasury = _fixedTaxForTreasury (acyc.sol#534)
Reentrancy in AllCoinsYieldCapital.transferFrom(address,address,uint256) (acyc.sol#194-209):
        External calls:
        - _transfer(sender,recipient,amount) (acyc.sol#199)
                - _treasuryAddress.transfer(amount) (acyc.sol#504)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (acyc.sol#200-207)
                - _allowances[owner][spender] = amount (acyc.sol#323)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (acyc.sol#324)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (acyc.sol#200-207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.
sol#10) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (@uniswap/v2-periphery/contracts/interfaces/IU
niswapV2Router01.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
AllCoinsYieldCapital._decimals (acyc.sol#58) should be constant
AllCoinsYieldCapital._name (acyc.sol#56) should be constant
AllCoinsYieldCapital._symbol (acyc.sol#57) should be constant
AllCoinsYieldCapital.uniDefault (acyc.sol#82) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.5.0 (@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol#1) allows old versions
Pragma version>=0.5.0 (@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#1) allows old versions
Pragma version>=0.6.2 (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#1) allows old versions
Pragma version>=0.6.2 (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol#1) allows old versions
Pragma version^0.8.0 (acyc.sol#38) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#18) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#19) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#36) is not in mixedCase
Function IUniswapV2Router01.WETH() (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#5) is not in mixedCase
Variable AllCoinsYieldCapital._treasuryAddress (acyc.sol#70) is not in mixedCase
Variable AllCoinsYieldCapital._fixedTaxForReflections (acyc.sol#73) is not in mixedCase
Variable AllCoinsYieldCapital._fixedTaxForTreasury (acyc.sol#74) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```


INFO:Detectors:
Different versions of Solidity is used:
- Version used: ['>=0.6.0', '>=0.6.2', '^0.8.0']
- ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
- ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/Address.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
- ^0.8.0 (@openzeppelin/contracts/utils/math/SafeMath.sol#4)
- >=0.5.0 (@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol#2)
- >=0.5.0 (@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#2)
- >=0.6.2 (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#2)
- >=0.6.2 (@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol#2)
- ^0.8.0 (acyc.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#90-92) is never used and should be removed
Address.functionCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#98-96) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#122-134) is never used and should be removed
Address.functionDelegateCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#27-37) is never used and should be removed
Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#178-216) is never used and should be removed
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeMath.div(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#161-168) is never used and should be removed
SafeMath.mod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#47-67) is never used and should be removed
SafeMath.trySub(uint256,uint256) (@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
AllCoinsYieldCapital._totalTokenSupply (acyc.sol#62) is set pre-construction with a non-constant function or state variable:
- 1 * 10 ** 12 * 10 ** _decimals
AllCoinsYieldCapital._totalReflections (acyc.sol#63) is set pre-construction with a non-constant function or state variable:
- (MAX - (MAX % _totalTokenSupply))
AllCoinsYieldCapital._minimumTokensToSwap (acyc.sol#68) is set pre-construction with a non-constant function or state variable:
- 10 * 10 ** 3 * 10 ** _decimals
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Reentrancy in AllCoinsYieldCapital._tokenTransfer(address,address,uint256) (acyc.sol#378-426):
External calls:
- _swapTokensForEth(contractTokenBalance) (acyc.sol#407)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(_treasuryAddress),block.timestamp) (acyc.sol#484-488)
External calls sending eth:
- _sendETHToTreasuryAddress(this.balance) (acyc.sol#411)
- _treasuryAddress.transfer(amount) (acyc.sol#504)
Event emitted after the call(s):
- Transfer(sender,recipient,reflectionsToCredit.div(_getRate())) (acyc.sol#425)
Reentrancy in AllCoinsYieldCapital.constructor(address,address) (acyc.sol#84-133):
External calls:
- _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (acyc.sol#111-113)
Event emitted after the call(s):
- Transfer(address(0),_msgSender(),_totalTokenSupply) (acyc.sol#132)
Reentrancy in AllCoinsYieldCapital.transferFrom(address,address,uint256) (acyc.sol#294-309):
External calls:
- _transfer(sender,recipient,amount) (acyc.sol#297)
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(_treasuryAddress),block.timestamp) (acyc.sol#484-488)
External calls sending eth:
- _transfer(sender,recipient,amount) (acyc.sol#297)
- _treasuryAddress.transfer(amount) (acyc.sol#504)
Event emitted after the call(s):
- Approval(owner,spender,amount) (acyc.sol#324)
- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (acyc.sol#298-307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
- INLINE ASM (@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage



Solhint Linter



Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

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

The audit showed several medium, low, and informational severity issues. In the end, the majority of the issues were acknowledged and fixed by the Auditee. All of the acknowledged issues shall be fixed in the next version of the contract; however, these issues have no bearing on the code's security.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the ACYC Token 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 ACYC Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
January, 2022

For

ACYC

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com