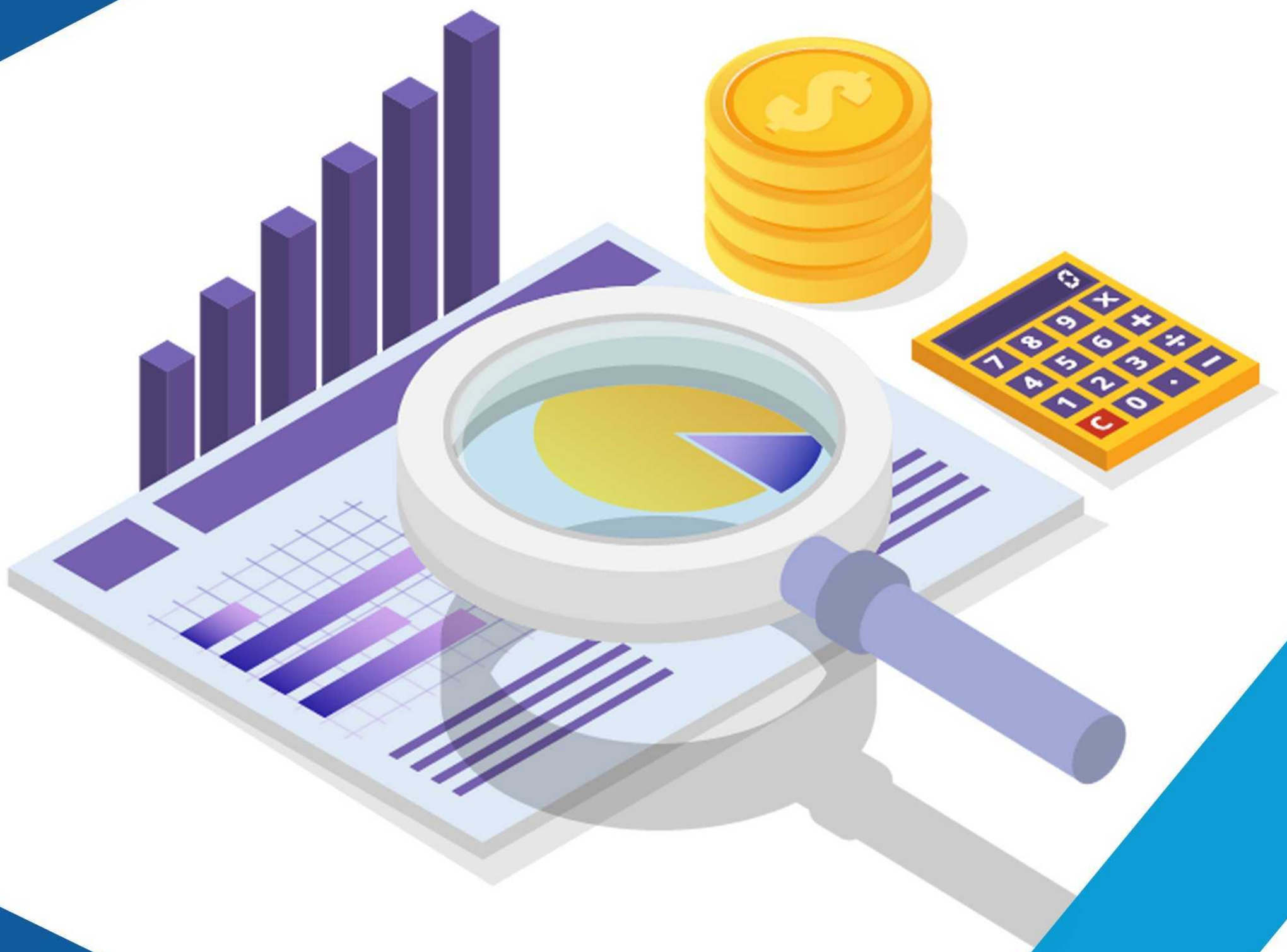
QuillAudits



Audit Report
July, 2021

REVOLT

Contents

Scope of Audit

The scope of this audit was to analyze and document the Revolt 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 analyzed 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.
SmartCheck.

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.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of 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 possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

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

High severity issues

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 to be fixed before moving to a live environment.

Medium level severity issues

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 level severity issues

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 four issues which 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	1	1	4
Closed	1	0	0	6

Introduction

During the period of **June 14, 2021 to June 19, 2021** - QuillAudits Team performed a security audit for Revolt smart contracts.

The code for the audit was taken from following the official link:
https://bscscan.com/address/0x87076dae0086d79873031e96a4b52c85f998ecb5#code

The rrr2.txt file was used for the final review.
(MD5 (rrr2.txt) = 1a376d844d44756f4a329598e9f8613e)
https://drive.google.com/file/d/1S8SAdAJ8O0_0hXDR7NnUnbPZnFrH78Xk/view?usp=sharing

Issues Found – Code Review / Manual Testing

High severity issues

1. Tax fee can be bypassed

Line	Code

793-798	function _getTValues(uint256 tAmount) private view returns (uint256, uint256, uint256) {
	uint256 tFee = tAmount.div(10 ** 2).mul(_taxFee);
	uint256 tLiquidity = tAmount.div(10 ** 2).mul(_liquidityFee);
	uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity);
	return (tTransferAmount, tFee, tLiquidity);
	}

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

In this case, the tFee is always Zero if the TAmount < 10**2. For example, with tAmount = 99; _taxFee = 5, tFee will be zero. If (tAmount * _taxFee / 10**2) was used, tFee would have been != 0.

In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

The users can bypass the tFee (without being charged by tFee) by just transferring 99 REVT tokens in each transaction.

Remediation

We recommend ordering multiplication before division at line 794 and 795:

```
uint256 tFee = tAmount.mul(_taxFee).div(10 ** 2);
uint256 tLiquidity = tAmount.mul(_liquidityFee).div(10 ** 2);
```

Status: Fixed

Medium severity issues

2. Missing Check for Reentrancy Attack

Line	Code

638	function transfer(address recipient, uint256 amount) public override returns (bool) {
	_transfer(_msgSender(), recipient, amount);
	return true;
	}

652	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;
	}

Description

Calling revolt.transfer() and revolt.transferFrom() might trigger function pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens() and pcsV2Router.addLiquidityETH() , which is implemented by third party at _pancakeswapV2Router . If there are vulnerable external calls in pcsV2Router , 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 pcsV2Router 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.

Remediation

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

Status: Unfixed

Low level severity issues

3. _excluded array is always empty

Line	Code

815	function _getCurrentSupply() private view returns(uint256, uint256) {
	uint256 rSupply = _rTotal;
	uint256 tSupply = _tTotal;
	for (uint256 i = 0; i < _excluded.length; i++) {
	if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
	rSupply = rSupply.sub(_rOwned[_excluded[i]]);
	tSupply = tSupply.sub(_tOwned[_excluded[i]]);
	}
	if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
	return (rSupply, tSupply);
	}

Description

The _excluded array is always Zero, which leads to the fact that the rSupply and tSupply variables will not be changed.

Remediation

We recommend removing the _getCurrentSupply() function and making _rTotal and _tTotal variables public, in case the _excluded array does not mean to contain any elements.

Status: Unfixed

Informational

4. Incorrect versions of Solidity & Different pragma directives are used

Pragma version>=0.6.2 (REVT.sol#341) allows old versions
Pragma version>=0.6.2 (REVT.sol#439) allows old versions
Pragma version>=0.5.0 (REVT.sol#482) allows old versions
Pragma version>=0.5.0 (REVT.sol#503) allows old versions

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.

Remediation
Use one Solidity version and deploy with the following Solidity versions: 0.6.12

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Status: Fixed

5. State variables that could be declared constant

_decimals
_maxTxAmount
_maxWalletToken
_name
_numTokensSellToAddToLiquidity
._symbol

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

Remediation
Add the constant attributes to state variables that never change.

Status: Unfixed

6. Variable Typos

Line	Code
280	tokensIntoLiqudity
302	contractBalanceRecepient

Description
There are typos in the above variables.

Remediation
We recommend correcting and changing tokensIntoLiqudity to tokensIntoLiquidity, contractBalanceRecepient to contractBalanceRecipient.

Status: Fixed

7. Wrong comments & initial values specified

Line	Code
280	uint256 private constant _tTotal = 40 * 10**9 * 10**9; // 50 Billion
302	uint256 public _maxTxAmount = 300 * 10**6 * 10**9; // Max Transaction: 200 Million (0.5%)
579	uint256 public _maxWalletToken = 600 * 10**6 * 10**9; // Max Wallet: 500 Million (1.5%)

Status: Partially fixed.
The values of _maxTxAmount and _maxTxAmount variables have been changed, but the comments have still remained.

Description
Wrong comments were found for the above code. As specified in the Smart Contract Specifications document, the Max Per Wallet is 500 Million, and the Max Per Transaction is 200 Million, but the initial values in the token are different.

A misunderstanding comment could influence code readability.

Remediation
We recommend correcting the values for _maxTxAmount and _maxTxAmount, or the comments for the above code should be corrected or removed properly.

8. Incorrect Error Message

Line	Code
732	require(contractBalanceRecepient + amount <= _maxWalletToken, "Exceeds maximum wallet token amount (100,000,000)");

Description
The error message in require(contractBalanceRecepient + amount <= _maxWalletToken, "Exceeds maximum wallet token amount (100,000,000)"); describe the error correctly.

Remediation
We recommend changing Exceeds maximum wallet token amount (100,000,000) to Exceeds maximum wallet token amount (600,000,000,000,000,000).

Status: Fixed

9. Conformance to Solidity naming conventions

Line	Code
732	uint256 private constant _tTotal = 40 * 10**9 * 10**9; // 50 Billion

Description
Constants should be named with all capital letters with underscores separating words. Examples: MAX_BLOCKS, TOKEN_NAME, TOKEN_TICKER, CONTRACT_VERSION

Remediation
Follow the Solidity naming convention.

Status: Unfixed

10. Conformance to Solidity naming conventions

Line	Code
585, 586	string private _name = 'Revolt'; string private _symbol = 'REVT';

Description
Single quote found in the above string variables. Whilst, the double quotes are being utilized for other string literals.

Remediation
We recommend using double quotes for string literals.

Status: Fixed

11. State Variable Default Visibility

Line	Code
570	bool inSwapAndLiquify;

Description
The Visibility of the inSwapAndLiquify variable is not defined. Labelling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

The default is internal for state variables, but it should be made explicit.

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

Variables can be specified as being public, internal, or private. Explicitly define visibility for all state variables.

Status: Unfixed

12. Order of Functions

Description

The following functions and function types should be re-ordered:

- The fallback function receive() external payable {}.
- The view and pure functions

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

Remediation

Functions should be grouped according to their visibility and ordered:

- constructor
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions last.

Status: Unfixed

13. Public function that could be declared external

Description

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

- reflect()
- reflectionFromToken()
- totalFees()

Remediation

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

Status: Unfixed

Functional test

Function Names	Testing results
transfer()	FAILED
transferFrom()	FAILED
increaseAllowance()	Passed
decreaseAllowance()	Passed
renounce()	Passed
reflect()	Passed
approve()	Passed

Automated Testing

Slither

```
enderphan@enderphan REVT % slither REVT.sol
INFO:Detectors:
Reentrancy in Revolt._transfer(address,address,uint256) (REVT.sol#720-771):
        External calls:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        State variables written after the call(s):
        - _transferStandard(sender,recipient,amount) (REVT.sol#767)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (REVT.sol#828)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (REVT.sol#775)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (REVT.sol#776)
        - _transferStandard(sender,recipient,amount) (REVT.sol#767)
                - _rTotal = _rTotal.sub(rFee) (REVT.sol#783)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
Revolt._tOwned (REVT.sol#562) is never initialized. It is used in:
        - Revolt._getCurrentSupply() (REVT.sol#813-823)
Revolt._excluded (REVT.sol#567) is never initialized. It is used in:
        - Revolt._getCurrentSupply() (REVT.sol#813-823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Revolt._getTValues(uint256) (REVT.sol#793-798) performs a multiplication on the result of a division:
        -tFee = tAmount.div(10 ** 2).mul(_taxFee) (REVT.sol#794)
Revolt._getTValues(uint256) (REVT.sol#793-798) performs a multiplication on the result of a division:
        -tLiquidity = tAmount.div(10 ** 2).mul(_liquidityFee) (REVT.sol#795)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Revolt.addLiquidity(uint256,uint256) (REVT.sol#873-886) ignores return value by pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,
0,0,owner(),block.timestamp) (REVT.sol#878-885)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Revolt.allowance(address,address).owner (REVT.sol#643) shadows:
        - Ownable.owner() (REVT.sol#299-301) (function)
Revolt._approve(address,address,uint256).owner (REVT.sol#697) shadows:
        - Ownable.owner() (REVT.sol#299-301) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
        - Revolt.allowance(address,address) (REVT.sol#643-645)
approve(address,uint256) should be declared external:
        - Revolt.approve(address,uint256) (REVT.sol#647-650)
transferFrom(address,address,uint256) should be declared external:
        - Revolt.transferFrom(address,address,uint256) (REVT.sol#652-656)
increaseAllowance(address,uint256) should be declared external:
        - Revolt.increaseAllowance(address,uint256) (REVT.sol#658-661)
decreaseAllowance(address,uint256) should be declared external:
        - Revolt.decreaseAllowance(address,uint256) (REVT.sol#663-666)
totalFees() should be declared external:
        - Revolt.totalFees() (REVT.sol#668-670)
reflect(uint256) should be declared external:
        - Revolt.reflect(uint256) (REVT.sol#672-678)
reflectionFromToken(uint256,bool) should be declared external:
        - Revolt.reflectionFromToken(uint256,bool) (REVT.sol#680-689)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (REVT.sol#350) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (REVT.sol#351)
Variable Revolt._transferStandard(address,address,uint256).rTransferAmount (REVT.sol#774) is too similar to Revolt._getTValues(uint256).tTransferAmount
(REVT.sol#796)
Variable Revolt.reflectionFromToken(uint256,bool).rTransferAmount (REVT.sol#686) is too similar to Revolt._getTValues(uint256).tTransferAmount (REVT.sol
#796)
Variable Revolt._getValues(uint256).rTransferAmount (REVT.sol#789) is too similar to Revolt._getTValues(uint256).tTransferAmount (REVT.sol#796)
Variable Revolt._transferStandard(address,address,uint256).rTransferAmount (REVT.sol#774) is too similar to Revolt._getValues(uint256).tTransferAmount (
REVT.sol#788)
Variable Revolt._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (REVT.sol#804) is too similar to Revolt._getValues(uint256).tTransferAmount
 (REVT.sol#788)
Variable Revolt._transferStandard(address,address,uint256).rTransferAmount (REVT.sol#774) is too similar to Revolt._transferStandard(address,address,uin
t256).tTransferAmount (REVT.sol#774)
Variable Revolt._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (REVT.sol#804) is too similar to Revolt._getTValues(uint256).tTransferAmoun
t (REVT.sol#796)
Variable Revolt._getValues(uint256).rTransferAmount (REVT.sol#789) is too similar to Revolt._getValues(uint256).tTransferAmount (REVT.sol#788)
Variable Revolt._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (REVT.sol#804) is too similar to Revolt._transferStandard(address,address,u
int256).tTransferAmount (REVT.sol#774)
Variable Revolt.reflectionFromToken(uint256,bool).rTransferAmount (REVT.sol#686) is too similar to Revolt._getValues(uint256).tTransferAmount (REVT.sol#
788)
Variable Revolt.reflectionFromToken(uint256,bool).rTransferAmount (REVT.sol#686) is too similar to Revolt._transferStandard(address,address,uint256).tTr
ansferAmount (REVT.sol#774)
Variable Revolt._getValues(uint256).rTransferAmount (REVT.sol#789) is too similar to Revolt._transferStandard(address,address,uint256).tTransferAmount (
REVT.sol#774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Revolt._decimals (REVT.sol#587) should be constant
Revolt._maxTxAmount (REVT.sol#578) should be constant
Revolt._maxWalletToken (REVT.sol#580) should be constant
Revolt._name (REVT.sol#585) should be constant
Revolt._numTokensSellToAddToLiquidity (REVT.sol#579) should be constant
Revolt._symbol (REVT.sol#586) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (REVT.sol#318-321)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (REVT.sol#327-329)
name() should be declared external:
        - Revolt.name() (REVT.sol#618-620)
symbol() should be declared external:
        - Revolt.symbol() (REVT.sol#622-624)
decimals() should be declared external:
        - Revolt.decimals() (REVT.sol#626-628)
totalSupply() should be declared external:
        - Revolt.totalSupply() (REVT.sol#630-632)
transfer(address,uint256) should be declared external:
        - Revolt.transfer(address,uint256) (REVT.sol#638-641)
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.5.0', '>=0.6.2', '^0.6.12']
        - ^0.6.12 (REVT.sol#6)
        - >=0.6.2 (REVT.sol#341)
        - >=0.6.2 (REVT.sol#439)
        - >=0.5.0 (REVT.sol#482)
        - >=0.5.0 (REVT.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (REVT.sol#115-118) is never used and should be removed
SafeMath.mod(uint256,uint256) (REVT.sol#249-251) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (REVT.sol#264-267) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Revolt._previousTaxFee (REVT.sol#576) is set pre-construction with a non-constant function or state variable:
        - _taxFee
Revolt._previousLiquidityFee (REVT.sol#577) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version>=0.6.2 (REVT.sol#341) allows old versions
Pragma version>=0.6.2 (REVT.sol#439) allows old versions
Pragma version>=0.5.0 (REVT.sol#482) allows old versions
Pragma version>=0.5.0 (REVT.sol#503) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IPancakeRouter01.WETH() (REVT.sol#345) is not in mixedCase
Function IPancakeFactory.INIT_CODE_PAIR_HASH() (REVT.sol#499) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (REVT.sol#520) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (REVT.sol#521) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (REVT.sol#538) is not in mixedCase
Constant Revolt._tTotal (REVT.sol#571) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Revolt._taxFee (REVT.sol#574) is not in mixedCase
Variable Revolt._liquidityFee (REVT.sol#575) is not in mixedCase
Variable Revolt._previousTaxFee (REVT.sol#576) is not in mixedCase
Variable Revolt._previousLiquidityFee (REVT.sol#577) is not in mixedCase
Variable Revolt._maxTxAmount (REVT.sol#578) is not in mixedCase
Variable Revolt._numTokensSellToAddToLiquidity (REVT.sol#579) is not in mixedCase
Variable Revolt._maxWalletToken (REVT.sol#580) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (REVT.sol#116)" inContext (REVT.sol#106-119)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (REVT.sol#350) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (REVT.sol#351)
Variable Revolt._transferStandard(address,address,uint256).rTransferAmount (REVT.sol#774) is too similar to Revolt._getTValues(uint256).tTransferAmount
(REVT.sol#796)
```

```
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (REVT.sol#654)
                - _allowances[owner][spender] = amount (REVT.sol#701)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Revolt._transfer(address,address,uint256) (REVT.sol#720-771):
        External calls:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (REVT.sol#779)
                - _transferStandard(sender,recipient,amount) (REVT.sol#767)
Reentrancy in Revolt.constructor() (REVT.sol#597-608):
        External calls:
        - pcsV2Pair = IPancakeFactory(_pancakeswapV2Router.factory()).createPair(address(this),_pancakeswapV2Router.WETH()) (REVT.sol#604)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (REVT.sol#607)
Reentrancy in Revolt.swapAndLiquify(uint256) (REVT.sol#831-852):
        External calls:
        - swapTokensForBNB(half) (REVT.sol#843)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (REVT.sol#702)
                - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
        - SwapAndLiquify(half,newBalance,otherHalf,contractTokenBalance) (REVT.sol#851)
Reentrancy in Revolt.transferFrom(address,address,uint256) (REVT.sol#652-656):
        External calls:
        - _transfer(sender,recipient,amount) (REVT.sol#653)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (REVT.sol#653)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (REVT.sol#702)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (REVT.sol#654)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Reentrancy in Revolt._transfer(address,address,uint256) (REVT.sol#720-771):
        External calls:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (REVT.sol#754)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        State variables written after the call(s):
        - removeAllFee() (REVT.sol#765)
                - _liquidityFee = 0 (REVT.sol#712)
        - restoreAllFee() (REVT.sol#770)
                - _liquidityFee = _previousLiquidityFee (REVT.sol#717)
        - removeAllFee() (REVT.sol#765)
                - _previousLiquidityFee = _liquidityFee (REVT.sol#709)
        - removeAllFee() (REVT.sol#765)
                - _previousTaxFee = _taxFee (REVT.sol#708)
        - _transferStandard(sender,recipient,amount) (REVT.sol#767)
                - _tFeeTotal = _tFeeTotal.add(tFee) (REVT.sol#784)
        - removeAllFee() (REVT.sol#765)
                - _taxFee = 0 (REVT.sol#711)
        - restoreAllFee() (REVT.sol#770)
                - _taxFee = _previousTaxFee (REVT.sol#716)
Reentrancy in Revolt.constructor() (REVT.sol#597-608):
        External calls:
        - pcsV2Pair = IPancakeFactory(_pancakeswapV2Router.factory()).createPair(address(this),_pancakeswapV2Router.WETH()) (REVT.sol#604)
        State variables written after the call(s):
        - pcsV2Router = _pancakeswapV2Router (REVT.sol#605)
Reentrancy in Revolt.swapAndLiquify(uint256) (REVT.sol#831-852):
        External calls:
        - swapTokensForBNB(half) (REVT.sol#843)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (REVT.sol#849)
                - _allowances[owner][spender] = amount (REVT.sol#701)
Reentrancy in Revolt.transferFrom(address,address,uint256) (REVT.sol#652-656):
        External calls:
        - _transfer(sender,recipient,amount) (REVT.sol#653)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
                - pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (REVT.sol#864-870)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (REVT.sol#653)
                - pcsV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (REVT.sol#878-885)
```

Mythril

```
Severity: Low
Contract: Revolt
Function name: constructor
PC address: 999
Estimated Gas Usage: 90244 - 501982
Multiple calls are executed in the same transaction.
This call is executed after a previous call in the same transaction. Try to isolate each call, transfer or send into its own transaction.
--------------------
In file: REVT.sol:604

_pancakeswapV2Router.WETH()

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0

==== External Call To Fixed Address ====
SWC ID: 107
Severity: Low
Contract: Revolt
Function name: constructor
PC address: 1166
Estimated Gas Usage: 86271 - 459751
The contract executes an external message call.
An external function call to a fixed contract address is executed. Make sure that the callee contract has been reviewed carefully.
--------------------
In file: REVT.sol:604

IPancakeFactory(_pancakeswapV2Router.factory()).createPair(address(this), _pancakeswapV2Router.WETH())

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
```

Manticore

```
enderphan@enderphan REVT % manticore REVT.sol --contract Revolt
2021-06-18 10:21:51,158: [24337] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2021-06-18 10:21:51,336: [24337] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOv
erflow, DetectUninitializedStorage, DetectUninitializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, Det
ectDelegatecall, DetectExternalCallAndLeak, DetectEnvInstruction, DetectManipulableBalance
2021-06-18 10:21:51,336: [24337] m.main:INFO: Beginning analysis
2021-06-18 10:21:51,341: [24337] m.e.manticore:INFO: Starting symbolic create contract
2021-06-18 10:21:58,104: [24337] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10' -> 12396694973890998440467380340983585058878106250672390494374587083972727727730
2021-06-18 10:22:13,795: [24337] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x11' -> 22581645139872629890233439717971975110198959689450188087151966948260709403752
2021-06-18 10:22:34,374: [24337] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe6\xff\xe1>/\x0c\xa
f2D\x01\x06#\x85Z\x1b\to\x88\xbc\x88\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x01' -> 92044731849928747291105575561355310160246347563324211658922605119408469872054
2021-06-18 10:22:34,925: [24337] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe6\xff\xe1>/\x0c\xa
f2D\x01\x06\n\xc82\x1e\xa0\xd3\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x04' -> 115104561533940427791757069509608498236630860181714524438378355443073614886581
2021-06-18 10:22:35,691: [24337] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00A\xcdp\xa2z3\x91\x0e\
xc5\x12w\xca\x04\x99\xbf\xcf\xa0\x0f\xf5\xdf\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x04' -> 3290796429592881489306671762322907901253846297319269808993654114566897853741
2021-06-18 10:22:54,321: [24337] m.e.manticore:INFO: Failed to create contract: exception in constructor
2021-06-18 10:22:54,368: [24364] m.c.manticore:INFO: Generated testcase No. 0 - REVERT(1 txs)
2021-06-18 10:22:54,369: [24364] m.c.plugin:WARNING: Caught will_solve in state None, but failed to capture its initialization
2021-06-18 10:22:54,401: [24363] m.c.manticore:INFO: Generated testcase No. 1 - REVERT(1 txs)
2021-06-18 10:22:54,402: [24363] m.c.plugin:WARNING: Caught will_solve in state None, but failed to capture its initialization
2021-06-18 10:22:54,705: [24337] m.c.manticore:INFO: Results in /Users/enderphan/Quilhash/REVT/mcore_ff5o5rug
```

Solhint Linter

Check-effects-interaction:

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

more

Pos: 596:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Revolt.swapTokensForBNB(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 854:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 868:12:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 883:12:

Solidity Static Analysis

Quillhash/REVT.sol:5:1: Error: Compiler version ^0.6.12 does not satisfy the r semver requirement
Quillhash/REVT.sol:108:27: Error: Code contains empty blocks
Quillhash/REVT.sol:340:1: Error: Compiler version >=0.6.2 does not satisfy the r semver requirement
Quillhash/REVT.sol:344:5: Error: Function name must be in mixedCase
Quillhash/REVT.sol:438:1: Error: Compiler version >=0.6.2 does not satisfy the r semver requirement
Quillhash/REVT.sol:481:1: Error: Compiler version >=0.5.0 does not satisfy the r semver requirement
Quillhash/REVT.sol:498:5: Error: Function name must be in mixedCase
Quillhash/REVT.sol:502:1: Error: Compiler version >=0.5.0 does not satisfy the r semver requirement
Quillhash/REVT.sol:519:5: Error: Function name must be in mixedCase
Quillhash/REVT.sol:520:5: Error: Function name must be in mixedCase
Quillhash/REVT.sol:537:5: Error: Function name must be in mixedCase
Quillhash/REVT.sol:556:1: Error: Contract has 18 states declarations but allowed no more than 15
Quillhash/REVT.sol:569:5: Error: Explicitly mark visibility of state
Quillhash/REVT.sol:570:30: Error: Constant name must be in capitalized SNAKE_CASE
Quillhash/REVT.sol:584:28: Error: Use double quotes for string literals
Quillhash/REVT.sol:585:30: Error: Use double quotes for string literals
Quillhash/REVT.sol:868:13: Error: Avoid to make time-based decisions in your business logic
Quillhash/REVT.sol:883:13: Error: Avoid to make time-based decisions in your business logic
Quillhash/REVT.sol:887:32: Error: Code contains empty blocks

Disclaimer

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

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

The audit showed several medium, low and informational severity issues. In the end, the majority of the issues were fixed or partially fixed by the Auditee. Other issues still remained unfixed as the internal team needs further discussion.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com