QuillAudits

Audit Report
June, 2021

Contents

Scope of Audit

The scope of this audit was to analyze and document the Moonship 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	3	3	5
Closed	1	0	0	0

Introduction

During the period of **June 11, 2021 to June 13, 2021** - QuillAudits Team performed a security audit for Moonship smart contracts.

The code for the audit was taken from following the official link:
https://github.com/moonshipfinance/contracts/blob/main/moonship.sol
Commit hash: 74658f3

Issues Found – Code Review / Manual Testing

High severity issues

1. Tax is applied when more than 0.01% of the total supply is sent

Line	Code

1325	function disruptiveTransfer(address recipient, uint256 amount) public payable returns (bool) {
	_transfer(_msgSender(), recipient, amount, msg.value);
	return true;
	}

Description:
According to HT Reward Pool agreements on moonship.finance, with respect to the Anti Pump-Dump-Exit Whales.

Transactions (sell/buy/transfer) that trade more than 0.01% of the total supply will be rejected. If whales don't want to be taxed 10% to make a transfer (between 2 wallets) that is larger than 0.01% of the total supply, they can use the Disruptive Transfer feature: the transfer will be charged for 2 HT without being taxed 10%.

This statement was tested and failed during the functional test. As a result, 10% of the tax was applied for the receiver when the sender sent more than 0.01% of the total of the supply along with 2 ETH.

POC:
1. The sender sent more than 0.01% of the total supply, which is 10e30 MSP (the sender is not an owner and 10e30 MSP = 0.1% of the total supply).
2. The receiver received less than 10e30 MSP (~90% of 10e30 MSP).

Expected result:
The receiver should have received the same amount of MSP sent by the sender without being taxed.

Remediation:
Please consider rewriting this function and add the sender to _isExcludedFromFee when the amount is > 0.01 of the total supply and msg.value is called.

Status: Closed
The team confirmed that 10% of the tax is still applied whenever the Disruptive Transfer feature is called by the Whales.

Medium severity issues

2. Missing Check for Reentrancy Attack

Line	Code

995	function transfer(address recipient, uint256 amount) public override returns (bool) {
	_transfer(_msgSender(), recipient, amount, 0);
	return true;
	}

1009	function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
	_transfer(sender, recipient, amount, 0);
	_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "BEP20: transfer amount exceeds allowance"));
	return true;
	}

Description:
Calling **moonship.transfer()** and **moonship.transferFrom()** might trigger function **nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens()** and **nutRouter.addLiquidityETH()** , which is implemented by third party at **InutRouter02** , in **Utils.swapTokensForEth()** and **Utils.addLiquidity()**. If there are vulnerable external calls in **InutRouter02** , 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 **InutRouter02** 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

Status: Acknowledged by the Auditee

3. Centralization Risks

Description:
The role owner has the authority to

- update settings
- manage the list containing contracts excluding from reward, fee, or max transaction limitation

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;
- Setting **_isExcludedFromMaxTx[owner()] = False**

Status: Acknowledged by the Auditee

4. Inappropriate Variable Initialization

Line	Code

1307	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(10000);
	}

Description:
_maxTxAmount should be 0.05% of the total supply according to the code comment.
But the **_maxTxPercent** can be set by the owner by calling **setMaxTxPercent** function at line 1307

Remediation:
We recommend setting the **_maxTxPercent** as a constant value which is calculated by 5% of the **_tTotal**

Status: Acknowledged by the Auditee

Low level severity issues

5. Tautology or Contradiction Issue

Line	Code
1340	require(balanceOf(msg.sender) >= 0, 'Error: must own MRAT to claim reward');

Description:
balanceOf(msg.sender) >= 0 is true even if the balance of the **msg.sender** is 0.
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 by the Auditee

6. Redundant Code

Line	Code
1245	else if (!_isExcluded[sender] && !_isExcluded[recipient])

Description:
When the contract enters the branch **else if (!_isExcluded[sender] && !_isExcluded[recipient])** or **else**, the contract will execute the same piece of **code _transferStandard(sender, recipient, amount)**;

Remediation:
We recommend removing the following code

Status: Acknowledged by the Auditee

7. Missing Range Check for Input Variable

Line	Code

1100	function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
	_taxFee = taxFee;
	}

1104	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
	_liquidityFee = liquidityFee;
	}

1307	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(10000);
	}

Description:
The role can set the following state variables arbitrary large or small causing potential risks in fees and anti whale :

- _taxFee
- _liquidityFee
- _maxTxAmount

Remediation:
We recommend setting ranges and check the following input variables:

- taxFee
- liquidityFee
- maxTxPercent

Status: Acknowledged by the Auditee

Informational

8. Variable Typo

Line	Code
933	uint256 tokensIntoLiqudity

Description:
There is a typo in **tokensIntoLiqudity**.

Remediation:
We recommend correcting and changing **tokensIntoLiqudity** to **tokensIntoLiquidity**.

Status: Acknowledged by the Auditee

9. Missing Events for Significant Transactions

Line	Code

1092	function excludeFromFee(address account) public onlyOwner {
	_isExcludedFromFee[account] = true;
	}

1906	function includeInFee(address account) public onlyOwner {
	_isExcludedFromFee[account] = false;
	}

1100	function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
	_taxFee = taxFee;
	}

1104	function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
	_liquidityFee = liquidityFee;
	}

1307	function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner() {
	_maxTxAmount = _tTotal.mul(maxTxPercent).div(10000);
	}

1311	function setExcludeFromMaxTx(address _address, bool value) public onlyOwner {
	_isExcludedFromMaxTx[_address] = value;
	}

Description:
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:

- setLiquidityFeePercent
- setTaxFeePercent
- includeInFee
- excludeFromFee
- setMaxTxPercent
- setExcludeFromMaxTx

Remediation:
We recommend emitting an event to log the update of the following variables:

- _maxTxAmount
- _isExcludedFromMaxTx
- _liquidityFee
- _taxFee
- _isExcludedFromFee

Status: Acknowledged by the Auditee

10. Inappropriate Location of Constant Declaration

Description:
A series of constants are declared in the middle of the contract.

Remediation:
We recommend declaring constants at the beginning of the contract.

Status: Acknowledged by the Auditee

11. Redundant Setting

Line	Code

1287,1288,
1291, 1296
	rewardCycleBlock = 7 days;
	easyRewardCycleBlock = 1 days;
	disruptiveCoverageFee = 2 ether;
	winningDoubleRewardPercentage = 5;

Description:
Variables have been set to appropriate values in the declaration, so there is no need to set them again.

Remediation:
We recommend removing redundant variable settings.

Status: Acknowledged by the Auditee

12. Incorrect versions of Solidity

Description:
Solidity version used: 0.6.8.
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:
Deploy with any of the following Solidity versions:

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6

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

Status: Acknowledged by the Auditee

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
transfer()	Passed
approve()	Passed
allowance()	Passed
transferFrom()	Passed
increaseAllowance()	Passed
decreaseAllowance()	Passed
totalFees()	Passed
deliver()	Passed

Function Names	Testing results
tokenFromReflection()	Passed
reflectionFromToken()	Passed
disruptiveTransfer()	FAILED
activateContract()	Passed
excludeFromReward()	Passed
includeInReward()	Passed
isExcludedFromReward()	Passed
setTaxFeePercent()	Passed
setLiquidityFeePercent()	Passed
setSwapAndLiquifyEnabled()	Passed
isExcludedFromFee()	Passed
setMaxTxPercent()	Passed
setExcludeFromMaxTx()	Passed
calculateBNBReward()	Passed
getRewardCycleBlock()	Passed
claimBNBReward()	Passed

Automated Testing

Slither

```
INFO:Detectors:
Utils.swapETHForTokens(address,address,uint256) (moonship.sol#785-804) sends eth to arbitrary user
        Dangerous calls:
        - nutRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,address(recipient),block.timestamp + 360) (moonship.sol#
798-803)
Utils.addLiquidity(address,address,uint256,uint256) (moonship.sol#806-823) sends eth to arbitrary user
        Dangerous calls:
        - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in MoonShip._transfer(address,address,uint256,uint256) (moonship.sol#1206-1231):
        External calls:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
                - nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (moonship.sol#776-782)
                - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
                - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        External calls sending eth:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (moonship.sol#1162)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (moonship.sol#1268)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (moonship.sol#1259)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (moonship.sol#1084)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (moonship.sol#1279)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (moonship.sol#1260)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (moonship.sol#1270)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (moonship.sol#1280)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (moonship.sol#1086)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _rTotal = _rTotal.sub(rFee) (moonship.sol#1117)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (moonship.sol#1164)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (moonship.sol#1083)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (moonship.sol#1278)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (moonship.sol#1269)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (moonship.sol#1085)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

```
        - MoonShip.decreaseAllowance(address,uint256) (moonship.sol#1020-1023)
isExcludedFromReward(address) should be declared external:
        - MoonShip.isExcludedFromReward(address) (moonship.sol#1025-1027)
totalFees() should be declared external:
        - MoonShip.totalFees() (moonship.sol#1029-1031)
deliver(uint256) should be declared external:
        - MoonShip.deliver(uint256) (moonship.sol#1033-1040)
reflectionFromToken(uint256,bool) should be declared external:
        - MoonShip.reflectionFromToken(uint256,bool) (moonship.sol#1042-1051)
excludeFromReward(address) should be declared external:
        - MoonShip.excludeFromReward(address) (moonship.sol#1059-1066)
excludeFromFee(address) should be declared external:
        - MoonShip.excludeFromFee(address) (moonship.sol#1092-1094)
includeInFee(address) should be declared external:
        - MoonShip.includeInFee(address) (moonship.sol#1096-1098)
isExcludedFromFee(address) should be declared external:
        - MoonShip.isExcludedFromFee(address) (moonship.sol#1194-1196)
setExcludeFromMaxTx(address,bool) should be declared external:
        - MoonShip.setExcludeFromMaxTx(address,bool) (moonship.sol#1311-1313)
claimBNBReward() should be declared external:
        - MoonShip.claimBNBReward() (moonship.sol#1338-1360)
disruptiveTransfer(address,uint256) should be declared external:
        - MoonShip.disruptiveTransfer(address,uint256) (moonship.sol#1390-1393)
activateContract() should be declared external:
        - MoonShip.activateContract() (moonship.sol#1448-1464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:moonship.sol analyzed (12 contracts with 75 detectors), 143 result(s) found
```

```
INFO:Detectors:
MoonShip._decimals (moonship.sol#922) should be constant
MoonShip._name (moonship.sol#920) should be constant
MoonShip._symbol (moonship.sol#921) should be constant
MoonShip._tTotal (moonship.sol#916) should be constant
MoonShip.rewardThreshold (moonship.sol#1303) should be constant
MoonShip.threshHoldTopUpRate (moonship.sol#1289) 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() (moonship.sol#442-445)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (moonship.sol#451-455)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (moonship.sol#457-459)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (moonship.sol#462-467)
unlock() should be declared external:
        - Ownable.unlock() (moonship.sol#470-475)
calculateBNBReward(uint256,uint256,uint256,uint256,uint256,address) should be declared external:
        - Utils.calculateBNBReward(uint256,uint256,uint256,uint256,uint256,address) (moonship.sol#710-736)
calculateTopUpClaim(uint256,uint256,uint256,uint256) should be declared external:
        - Utils.calculateTopUpClaim(uint256,uint256,uint256,uint256) (moonship.sol#738-762)
swapTokensForEth(address,uint256) should be declared external:
        - Utils.swapTokensForEth(address,uint256) (moonship.sol#764-783)
swapETHForTokens(address,address,uint256) should be declared external:
        - Utils.swapETHForTokens(address,address,uint256) (moonship.sol#785-804)
addLiquidity(address,address,uint256,uint256) should be declared external:
        - Utils.addLiquidity(address,address,uint256,uint256) (moonship.sol#806-823)
name() should be declared external:
        - MoonShip.name() (moonship.sol#974-976)
symbol() should be declared external:
        - MoonShip.symbol() (moonship.sol#978-980)
decimals() should be declared external:
        - MoonShip.decimals() (moonship.sol#982-984)
totalSupply() should be declared external:
        - MoonShip.totalSupply() (moonship.sol#986-988)
transfer(address,uint256) should be declared external:
        - MoonShip.transfer(address,uint256) (moonship.sol#995-998)
allowance(address,address) should be declared external:
        - MoonShip.allowance(address,address) (moonship.sol#1000-1002)
approve(address,uint256) should be declared external:
        - MoonShip.approve(address,uint256) (moonship.sol#1004-1007)
transferFrom(address,address,uint256) should be declared external:
        - MoonShip.transferFrom(address,address,uint256) (moonship.sol#1009-1013)
increaseAllowance(address,uint256) should be declared external:
        - MoonShip.increaseAllowance(address,uint256) (moonship.sol#1015-1018)
decreaseAllowance(address,uint256) should be declared external:
```

```
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._transferStandard(address,address,uint256).tTransfe
rAmount (moonship.sol#1258)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._transferStandard(address,address,uin
t256).tTransferAmount (moonship.sol#1258)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._getValues(uint256).tTransferAmount (
moonship.sol#1122)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._transferBothExcluded(addres
s,address,uint256).tTransferAmount (moonship.sol#1082)
Variable MoonShip._transferFromExcluded(address,address,uint256).rTransferAmount (moonship.sol#1277) is too similar to MoonShip._transferToExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1267)
Variable MoonShip._transferFromExcluded(address,address,uint256).rTransferAmount (moonship.sol#1277) is too similar to MoonShip._transferToExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1267)
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._getTValues(uint256).tTransferAmount (moonship.sol#
1130)
Variable MoonShip._transferFromExcluded(address,address,uint256).rTransferAmount (moonship.sol#1277) is too similar to MoonShip._transferStandard(addres
s,address,uint256).tTransferAmount (moonship.sol#1258)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._transferToExcluded(address,
address,uint256).tTransferAmount (moonship.sol#1267)
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._transferFromExcluded(address,address,uint256).tTra
nsferAmount (moonship.sol#1277)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._transferFromExcluded(address,address
,uint256).tTransferAmount (moonship.sol#1277)
Variable MoonShip._transferFromExcluded(address,address,uint256).rTransferAmount (moonship.sol#1277) is too similar to MoonShip._transferBothExcluded(ad
dress,address,uint256).tTransferAmount (moonship.sol#1082)
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._getValues(uint256).tTransferAmount (moonship.sol#1
122)
Variable MoonShip._transferBothExcluded(address,address,uint256).rTransferAmount (moonship.sol#1082) is too similar to MoonShip._transferToExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1267)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._transferToExcluded(address,address,u
int256).tTransferAmount (moonship.sol#1267)
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._transferToExcluded(address,address,uint256).tTrans
ferAmount (moonship.sol#1267)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._getTValues(uint256).tTransferAmount
(moonship.sol#1130)
Variable MoonShip._getValues(uint256).rTransferAmount (moonship.sol#1123) is too similar to MoonShip._transferBothExcluded(address,address,uint256).tTra
nsferAmount (moonship.sol#1082)
Variable MoonShip.reflectionFromToken(uint256,bool).rTransferAmount (moonship.sol#1048) is too similar to MoonShip._transferBothExcluded(address,address
,uint256).tTransferAmount (moonship.sol#1082)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
MoonShip.constructor(address) (moonship.sol#948-972) uses literals with too many digits:
        - _isExcludedFromMaxTx[address(0x000000000000000000000000000000000000dEaD)] = true (moonship.sol#968)
MoonShip.calculateBNBReward(address) (moonship.sol#1315-1331) uses literals with too many digits:
        - tSupply = uint256(_tTotal).sub(balanceOf(address(0))).sub(balanceOf(0x000000000000000000000000000000000000dEaD)).sub(balanceOf(0x0d15E114cdD9F
3f2B3bc0De19D5C9c5e4994ee8F)).sub(balanceOf(address(nutPair))) (moonship.sol#1316-1320)
MoonShip.claimBNBReward() (moonship.sol#1338-1360) uses literals with too many digits:
        - Utils.swapETHForTokens(address(nutRouter),address(0x000000000000000000000000000000000000dEaD),reward.div(5)) (moonship.sol#1346-1350)
MoonShip.slitherConstructorVariables() (moonship.sol#901-1466) uses literals with too many digits:
        - _tTotal = 1000000000 * 10 ** 6 * 10 ** 18 (moonship.sol#916)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
Parameter MoonShip.calculateLiquidityFee(uint256)._amount (moonship.sol#1173) is not in mixedCase
Parameter MoonShip.setExcludeFromMaxTx(address,bool)._address (moonship.sol#1311) is not in mixedCase
Variable MoonShip._maxTxAmount (moonship.sol#1290) is not in mixedCase
Variable MoonShip._taxFee (moonship.sol#1298) is not in mixedCase
Variable MoonShip._liquidityFee (moonship.sol#1301) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (moonship.sol#248)" inContext (moonship.sol#242-251)
Redundant expression "_tTotal (moonship.sol#732)" inUtils (moonship.sol#685-824)
Redundant expression "ofAddress (moonship.sol#733)" inUtils (moonship.sol#685-824)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable InutRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (moonship.sol#552) is too similar to
InutRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (moonship.sol#553)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._transferBothExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1082)
Variable MoonShip._transferBothExcluded(address,address,uint256).rTransferAmount (moonship.sol#1082) is too similar to MoonShip._getTValues(uint256).tTr
ansferAmount (moonship.sol#1130)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._transferStandard(address,ad
dress,uint256).tTransferAmount (moonship.sol#1258)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._transferStandard(address,
address,uint256).tTransferAmount (moonship.sol#1258)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._transferStandard(address,
address,uint256).tTransferAmount (moonship.sol#1258)
Variable MoonShip._transferBothExcluded(address,address,uint256).rTransferAmount (moonship.sol#1082) is too similar to MoonShip._transferBothExcluded(ad
dress,address,uint256).tTransferAmount (moonship.sol#1082)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._transferFromExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1277)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._getTValues(uint256).tTransf
erAmount (moonship.sol#1130)
Variable MoonShip._transferBothExcluded(address,address,uint256).rTransferAmount (moonship.sol#1082) is too similar to MoonShip._transferStandard(addres
s,address,uint256).tTransferAmount (moonship.sol#1258)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._getValues(uint256).tTrans
ferAmount (moonship.sol#1122)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._getValues(uint256).tTransfe
rAmount (moonship.sol#1122)
Variable MoonShip._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (moonship.sol#1138) is too similar to MoonShip._getValues(uint256).tTrans
ferAmount (moonship.sol#1122)
Variable MoonShip._transferStandard(address,address,uint256).rTransferAmount (moonship.sol#1258) is too similar to MoonShip._transferFromExcluded(addres
s,address,uint256).tTransferAmount (moonship.sol#1277)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._transferToExcluded(addres
s,address,uint256).tTransferAmount (moonship.sol#1267)
Variable MoonShip._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (moonship.sol#1138) is too similar to MoonShip._transferFromExcluded(addr
ess,address,uint256).tTransferAmount (moonship.sol#1277)
Variable MoonShip._transferFromExcluded(address,address,uint256).rTransferAmount (moonship.sol#1277) is too similar to MoonShip._transferFromExcluded(ad
dress,address,uint256).tTransferAmount (moonship.sol#1277)
Variable MoonShip._transferToExcluded(address,address,uint256).rTransferAmount (moonship.sol#1267) is too similar to MoonShip._getTValues(uint256).tTran
sferAmount (moonship.sol#1130)
Variable MoonShip._transferBothExcluded(address,address,uint256).rTransferAmount (moonship.sol#1082) is too similar to MoonShip._getValues(uint256).tTra
nsferAmount (moonship.sol#1122)
Variable MoonShip._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (moonship.sol#1138) is too similar to MoonShip._transferStandard(address,
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
MoonShip.ensureMaxTxAmount(address,address,uint256,uint256) (moonship.sol#1374-1388) compares to a boolean constant:
        -_isExcludedFromMaxTx[from] == false && _isExcludedFromMaxTx[to] == false (moonship.sol#1381-1382)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (moonship.sol#368-389) is never used and should be removed
Address.functionCall(address,bytes) (moonship.sol#328-330) is never used and should be removed
Address.functionCall(address,bytes,string) (moonship.sol#338-340) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (moonship.sol#353-355) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (moonship.sol#363-366) is never used and should be removed
Address.isContract(address) (moonship.sol#275-284) is never used and should be removed
Address.sendValue(address,uint256) (moonship.sol#302-308) is never used and should be removed
Context._msgData() (moonship.sol#247-250) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
MoonShip._rTotal (moonship.sol#917) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
MoonShip._maxTxAmount (moonship.sol#1290) is set pre-construction with a non-constant function or state variable:
        - _tTotal
MoonShip._previousTaxFee (moonship.sol#1299) is set pre-construction with a non-constant function or state variable:
        - _taxFee
MoonShip._previousLiquidityFee (moonship.sol#1302) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
MoonShip.minTokenNumberToSell (moonship.sol#1305) is set pre-construction with a non-constant function or state variable:
        - _tTotal.mul(1).div(10000).div(10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version>=0.6.8 (moonship.sol#12) allows old versions
Pragma version>=0.6.8 (moonship.sol#682) allows old versions
Pragma version>=0.6.8 (moonship.sol#828) allows old versions
Pragma version>=0.6.8 (moonship.sol#896) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (moonship.sol#302-308):
        - (success) = recipient.call{value: amount}() (moonship.sol#306)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (moonship.sol#368-389):
        - (success,returndata) = target.call{value: weiValue}(data) (moonship.sol#372)
Low level call in MoonShip.claimBNBReward() (moonship.sol#1338-1360):
        - (sent) = address(msg.sender).call{value: reward}() (moonship.sol#1358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function InutPair.DOMAIN_SEPARATOR() (moonship.sol#509) is not in mixedCase
Function InutPair.PERMIT_TYPEHASH() (moonship.sol#510) is not in mixedCase
Function InutPair.MINIMUM_LIQUIDITY() (moonship.sol#527) is not in mixedCase
Function InutRouter01.WETH() (moonship.sol#547) is not in mixedCase
Parameter MoonShip.setSwapAndLiquifyEnabled(bool)._enabled (moonship.sol#1108) is not in mixedCase
Parameter MoonShip.calculateTaxFee(uint256)._amount (moonship.sol#1167) is not in mixedCase
Parameter MoonShip.calculateLiquidityFee(uint256)._amount (moonship.sol#1173) is not in mixedCase
```

```
        - ClaimBNBSuccessfully(msg.sender,reward,nextAvailableClaimDate[msg.sender]) (moonship.sol#1358)
Reentrancy in MoonShip.constructor(address) (moonship.sol#948-972):
        External calls:
        - nutPair = IPancakeFactory(_nutRouter.factory()).createPair(address(this),_nutRouter.WETH()) (moonship.sol#955-956)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (moonship.sol#971)
Reentrancy in MoonShip.swapAndLiquify(address,address) (moonship.sol#1395-1446):
        External calls:
        - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
        - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        Event emitted after the call(s):
        - SwapAndLiquify(piece,deltaBalance,otherPiece) (moonship.sol#1444)
Reentrancy in MoonShip.transferFrom(address,address,uint256) (moonship.sol#1009-1013):
        External calls:
        - _transfer(sender,recipient,amount,0) (moonship.sol#1010)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
                - nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (moonship.sol#776-782)
                - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
                - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        External calls sending eth:
        - _transfer(sender,recipient,amount,0) (moonship.sol#1010)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (moonship.sol#1203)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (moonship.sol#101
1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (moonship.sol#470-475) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked until 7 days) (moonship.sol#472)
Utils.isLotteryWon(uint256,uint256) (moonship.sol#704-708) uses timestamp for comparisons
        Dangerous comparisons:
        - luckyNumber <= winPercentage (moonship.sol#707)
MoonShip.getRewardCycleBlock() (moonship.sol#1333-1336) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= disableEasyRewardFrom (moonship.sol#1334)
MoonShip.claimBNBReward() (moonship.sol#1338-1360) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(nextAvailableClaimDate[msg.sender] <= block.timestamp,Error: next available not reached) (moonship.sol#1339)
MoonShip.ensureMaxTxAmount(address,address,uint256,uint256) (moonship.sol#1374-1388) uses timestamp for comparisons
        Dangerous comparisons:
        - value < disruptiveCoverageFee && block.timestamp >= disruptiveTransferEnabledFrom (moonship.sol#1384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (moonship.sol#275-284) uses assembly
        - INLINE ASM (moonship.sol#282)
Address._functionCallWithValue(address,bytes,uint256,string) (moonship.sol#368-389) uses assembly
```

```
ycleBlock,threshHoldTopUpRate,amount) (moonship.sol#1366-1371)
Reentrancy in MoonShip.constructor(address) (moonship.sol#948-972):
        External calls:
        - nutPair = IPancakeFactory(_nutRouter.factory()).createPair(address(this),_nutRouter.WETH()) (moonship.sol#955-956)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (moonship.sol#962)
        - _isExcludedFromFee[address(this)] = true (moonship.sol#963)
        - _isExcludedFromMaxTx[owner()] = true (moonship.sol#966)
        - _isExcludedFromMaxTx[address(this)] = true (moonship.sol#967)
        - _isExcludedFromMaxTx[address(0x000000000000000000000000000000000000dEaD)] = true (moonship.sol#968)
        - _isExcludedFromMaxTx[address(0)] = true (moonship.sol#969)
        - nutRouter = _nutRouter (moonship.sol#959)
Reentrancy in MoonShip.transferFrom(address,address,uint256) (moonship.sol#1009-1013):
        External calls:
        - _transfer(sender,recipient,amount,0) (moonship.sol#1010)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
                - nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (moonship.sol#776-782)
                - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
                - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        External calls sending eth:
        - _transfer(sender,recipient,amount,0) (moonship.sol#1010)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (moonship.sol#1011)
                - _allowances[owner][spender] = amount (moonship.sol#1202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MoonShip._transfer(address,address,uint256,uint256) (moonship.sol#1206-1231):
        External calls:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
                - nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (moonship.sol#776-782)
                - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
                - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        External calls sending eth:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (moonship.sol#1263)
                - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
        - Transfer(sender,recipient,tTransferAmount) (moonship.sol#1273)
                - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
        - Transfer(sender,recipient,tTransferAmount) (moonship.sol#1283)
                - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
        - Transfer(sender,recipient,tTransferAmount) (moonship.sol#1089)
                - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
Reentrancy in MoonShip.claimBNBReward() (moonship.sol#1338-1360):
        External calls:
        - Utils.swapETHForTokens(address(nutRouter),address(0x000000000000000000000000000000000000dEaD),reward.div(5)) (moonship.sol#1346-1350)
```

```
INFO:Detectors:
Reentrancy in MoonShip.claimBNBReward() (moonship.sol#1338-1360):
        External calls:
        - Utils.swapETHForTokens(address(nutRouter),address(0x000000000000000000000000000000000000dEaD),reward.div(5)) (moonship.sol#1346-1350)
        State variables written after the call(s):
        - nextAvailableClaimDate[msg.sender] = block.timestamp + getRewardCycleBlock() (moonship.sol#1355)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
MoonShip.claimBNBReward() (moonship.sol#1338-1360) contains a tautology or contradiction:
        - require(bool,string)(balanceOf(msg.sender) >= 0,Error: must own MRAT to claim reward) (moonship.sol#1340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
Utils.addLiquidity(address,address,uint256,uint256) (moonship.sol#806-823) ignores return value by nutRouter.addLiquidityETH{value: ethAmount}(address(t
his),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
MoonShip.allowance(address,address).owner (moonship.sol#1000) shadows:
        - Ownable.owner() (moonship.sol#423-425) (function)
MoonShip._approve(address,address,uint256).owner (moonship.sol#1198) shadows:
        - Ownable.owner() (moonship.sol#423-425) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in MoonShip._transfer(address,address,uint256,uint256) (moonship.sol#1206-1231):
        External calls:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
                - nutRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (moonship.sol#776-782)
                - Utils.swapTokensForEth(address(nutRouter),tokenAmountToBeSwapped) (moonship.sol#1429)
                - Utils.addLiquidity(address(nutRouter),owner(),otherPiece,bnbToBeAddedToLiquidity) (moonship.sol#1442)
        External calls sending eth:
        - swapAndLiquify(from,to) (moonship.sol#1219)
                - nutRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner,block.timestamp + 360) (moonship.sol#815-822)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _liquidityFee = _previousLiquidityFee (moonship.sol#1191)
                - _liquidityFee = 0 (moonship.sol#1186)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _previousLiquidityFee = _liquidityFee (moonship.sol#1183)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _previousTaxFee = _taxFee (moonship.sol#1182)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _tFeeTotal = _tFeeTotal.add(tFee) (moonship.sol#1118)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - _taxFee = _previousTaxFee (moonship.sol#1190)
                - _taxFee = 0 (moonship.sol#1185)
        - _tokenTransfer(from,to,amount,takeFee) (moonship.sol#1230)
                - nextAvailableClaimDate[recipient] = nextAvailableClaimDate[recipient] + Utils.calculateTopUpClaim(currentRecipientBalance,basedRewardC
```

Mythril

```
enderphan@enderphan Downloads % myth a moonship.sol
The analysis was completed successfully. No issues were detected.
```

Solhint Linter

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases.
If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 64:16:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 368:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MoonShip.(address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 883:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 282:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 381:16:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp".
"block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 465:20:

Solhint

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: 1385:32:

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: 1393:40:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 306:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 372:50:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1293:23:

Solidity Static Analysis

- Quillhash/moonship.sol:833:1: Error: Import statements must be on top
- Quillhash/moonship.sol:834:1: Error: Import statements must be on top
- Quillhash/moonship.sol:12:1: Error: Compiler version >=0.6.8 does not satisfy the r semver requirement
- Quillhash/moonship.sol:465:21: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:472:17: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:509:5: Error: Function name must be in mixedCase
- Quillhash/moonship.sol:510:5: Error: Function name must be in mixedCase
- Quillhash/moonship.sol:527:5: Error: Function name must be in mixedCase
- Quillhash/moonship.sol:547:5: Error: Function name must be in mixedCase
- Quillhash/moonship.sol:682:1: Error: Compiler version >=0.6.8 does not satisfy the r semver requirement
- Quillhash/moonship.sol:692:21: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:693:80: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:695:76: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:745:20: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:781:13: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:802:13: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:821:13: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:830:1: Error: Compiler version >=0.6.8 does not satisfy the r semver requirement
- Quillhash/moonship.sol:836:1: Error: Contract has 30 states declarations

- Quillhash/moonship.sol:1269:13: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:1273:52: Error: Visibility modifier must be first in list of modifiers
- Quillhash/moonship.sol:1274:55: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:1274:72: Error: Use double quotes for string literals
- Quillhash/moonship.sol:1275:45: Error: Use double quotes for string literals
- Quillhash/moonship.sol:1290:46: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:1293:24: Error: Avoid to use low level calls.
- Quillhash/moonship.sol:1294:23: Error: Use double quotes for string literals
- Quillhash/moonship.sol:1319:50: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:1385:33: Error: Avoid to make time-based decisions in your business logic
- Quillhash/moonship.sol:1393:41: Error: Avoid to make time-based decisions in your business logic

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.
Numerous issues were discovered during the audit. It is recommended to kindly go through the above-mentioned details and fix the code accordingly.

Disclaimer

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

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com