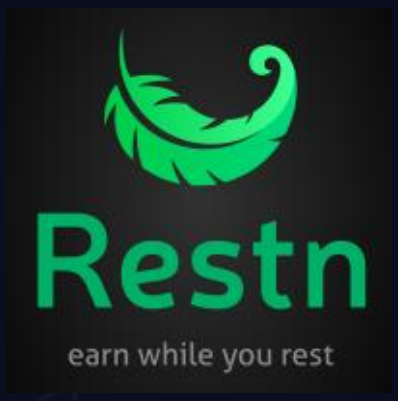
expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

Restn Fair Launch



Contract Address

0xc4fAF18Bc0928a42c66fDAb7A8498e33f6e69247

Audit Summary

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED**

Ownership: NOT **RENOUNCED**

KYC Verification: Done

Audit Date: 24/07/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

Contract Review

Contract Name	**Restn**
Compiler Version	**v0.7.4+commit.3f05b770**
Optimization	**Yes with 200 runs**
License	**MIT license**
Explorer	**https://bscscan.com/address/0xc4fAF18Bc0928a42c66fDAb7A8498e33f6e69247#contracts**
Symbol	**RST**
Decimals	**18**
Total Supply	**100,000,000**
Domain	**https://restn.org/**

Project Review

Token Name: Restn

Web Site: https://restn.org/

Twitter: https://twitter.com/RestnOfficial

Telegram: https://t.me/RestnOfficial

Contract Address:
0xc4fAF18Bc0928a42c66fDAb7A8498e33f6e69247

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

Low-Risk

3 low-risk code issues found

Medium-Risk

0 medium-risk code issues found

High-Risk

0 high-risk code issues found

Audit Summary

```
Compiled with solc
Number of lines: 939 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 11 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 15
Number of informational issues: 57
Number of low issues: 3
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20
```

Name	# functions	ERCS	ERC20 info	Complex code	Features
SafeMathInt	6			No	
SafeMath	7			No	
InterfaceLP	1			No	
Roles	3			No	
IDEXRouter	7			No	Receive ETH
IDEXFactory	1			No	
Restn	74	ERC20	No Minting Approve Race Cond.	Yes	Receive ETH Send ETH Tokens interaction

Low-Risk

1) Contract contains Reentrancy vulnuerabilities

```
function _transferFrom(address sender, address recipient, uint256 amount) internal returns (bool) {
        bool excludedAccount = _isFeeExempt[sender] || _isFeeExempt[recipient];

        require(tradingActive || excludedAccount, "Trading not started");
        require(!blacklist[sender] && !blacklist[recipient], "in_blacklist");

        if (!excludedAccount && !automatedMarketMakerPairs[sender]){
            uint _day = getDay();
            if (_day != tradeData[sender].day){
                tradeData[sender].day = _day;
                tradeData[sender].numberOfTrades = 0;
            }
            require (tradeData[sender].numberOfTrades <= numberOfTradesAllowed, "You exceed the number of trades
allowed");
            tradeData[sender].numberOfTrades = (tradeData[sender].numberOfTrades).add(1);
        }

        if (inSwap) {
            return _basicTransfer(sender, recipient, amount);
        }

        uint256 gonAmount = amount.mul(_gonsPerFragment);

        if (shouldSwapBack()) {
            swapBack();
        }

        _gonBalances[sender] = _gonBalances[sender].sub(gonAmount);

        uint256 gonAmountReceived = shouldTakeFee(sender, recipient) ? takeFee(sender, recipient, gonAmount) :
gonAmount;
        _gonBalances[recipient] = _gonBalances[recipient].add(gonAmountReceived);
```

Recommendation

Apply the check-effects-interaction pattern

2) Local variable shadowing

Detection of shadowing using local variables.

```
Constructor() ERC20Detailed("Restn", "RST", uint8(DECIMALS)) Ownable() {
        router = IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
        pair = IDEXFactory(router.factory()).createPair(address(this), router.WETH());
        address pairBusd = IDEXFactory(router.factory()).createPair(address(this), busdToken);
        address _owner = 0x92984B9771Baa7e084Ac12D312370103e9261235;
```

Recommendation

Rename the local variables that shadow another component.

3) Unused return

The return value of an external call is not stored in a local or state variable.

```
constructor() ERC20Detailed("Restn", "RST", uint8(DECIMALS)) Ownable() {
        router = IDEXRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
        pair = IDEXFactory(router.factory()).createPair(address(this), router.WETH());
        address pairBusd = IDEXFactory(router.factory()).createPair(address(this), busdToken);
        address _owner = 0x92984B9771Baa7e084Ac12D312370103e9261235;

        _allowedFragments[address(this)][address(router)] = uint256(-1);
        _allowedFragments[address(this)][pair] = uint256(-1);
        _allowedFragments[address(this)][address(this)] = uint256(-1);
        _allowedFragments[address(this)][pairBusd] = uint256(-1);

        setAutomatedMarketMakerPair(pair, true);
        setAutomatedMarketMakerPair(pairBusd, true);

        _totalSupply = INITIAL_FRAGMENTS_SUPPLY;
        _gonBalances[_owner] = TOTAL_GONS;
        _gonsPerFragment = TOTAL_GONS.div(_totalSupply);

        _isFeeExempt[treasuryReceiver] = true;
        _isFeeExempt[riskFreeValueReceiver] = true;
        _isFeeExempt[address(this)] = true;
        _isFeeExempt[_owner] = true;

        IERC20(busdToken).approve(address(router), uint256(-1));
        IERC20(busdToken).approve(address(pairBusd), uint256(-1));
        IERC20(busdToken).approve(address(this), uint256(-1));

        _transferOwnership(_owner);

        emit Transfer(address(0x0), _owner, _totalSupply);
    }
```

Recommendation

Ensure that all the return values of function calls are used.

Manual Audit (Contract Function)

```
contract Restn is ERC20Detailed, Ownable, WhitelistedRole {
    using SafeMath for uint256;
    using SafeMathInt for int256;

    bool public tradingActive = false;
    bool public swapEnabled = false;
    bool public autoRebase = false;
    bool public feesOnNormalTransfers = false;
    bool public isLiquidityInBnb = true;

    uint256 public rewardYield = 1049125;
    uint256 public rewardYieldDenominator = 10000000000;

    uint256 public rebaseFrequency = 1800;
    uint256 public nextRebase = block.timestamp + 604800;

    mapping(address => bool) _isFeeExempt;
    address[] public _markerPairs;
    mapping (address => bool) public automatedMarketMakerPairs;
    mapping(address => bool) public blacklist;


    uint256 private constant MAX_REBASE_FREQUENCY = 1800;
    uint256 private constant DECIMALS = 18;
    uint256 private constant MAX_UINT256 = ~uint256(0);
    uint256 private constant INITIAL_FRAGMENTS_SUPPLY = 1 * 10**8 * 10**DECIMALS;
    uint256 private constant TOTAL_GONS = MAX_UINT256 - (MAX_UINT256 % INITIAL_FRAGMENTS_SUPPLY);
    uint256 private constant MAX_SUPPLY = 3 * 10**9 * 10**DECIMALS;

    address DEAD = 0x000000000000000000000000000000000000dEaD;
    address ZERO = 0x0000000000000000000000000000000000000000;

    address public liquidityReceiver = DEAD;
    address public treasuryReceiver = 0xB88b72Aa8f1883dF85E84258E933002E28923d0D;
    address public riskFreeValueReceiver = 0x04A8A6d4c27f8707488FF97654e0D41398303133;
    address public busdToken = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;

    IDEXRouter public router;
    address public pair;

    uint256 public liquidityFee = 20;
    uint256 public burnFee = 10;
    uint256 public treasuryFee = 30;
    uint256 public RFVFee = 40;
```

Important Points To Consider

✓ Verified contract source

✗ Token is sellable (not a honeypot) at this time

✗ Ownership renounced or source does not contain an owner contract

✗ Source does not contain a fee modifier

✓ Owner/creator wallet contains less than 10% of circulating token supply (4.99%)

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co