BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT

Project: Rayons Energy

Website: https://rayonsenergy.com/

PASSED

BlockSAFU Score:

97

Contract Address:

0xDe95A749Cbb93940A06c68FfA78ECFce1b1F7d9a

Disclamer: BlockSAFU is not responsible for any financial losses.
Nothing in this contract audit is financial advice, please do your own reasearch.

DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur to the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 13% (owner can’t set fees over 25%).
- Sell 13% (owner can’t set fees over 25%).

Tx Amount

- Owner can set max tx amount, but with limitation, can’t be set lower than 0.1% of the total supply.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner can’t set blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner can limit the number of wallet holdings but with limitation, can’t be set lower than 0.1% from total supply.

Trading Cooldown

- Owner can set the selling time interval, but with limitation, can’t set more than 1 hour.

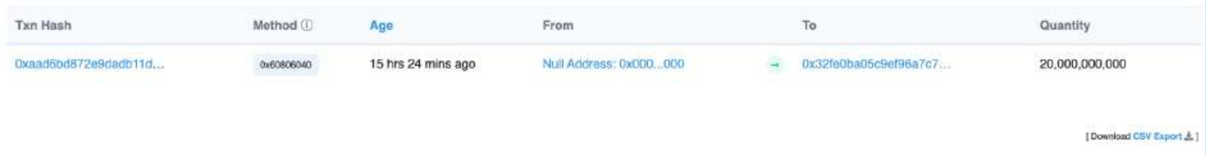
SMART CONTRACT REVIEW

Token Name	**RYS**
Token Symbol	**$RYS**
Token Decimal	18
Total Supply	20,000,000,000 **$RYS**
Contract Address	0xDe95A749Cbb93940A06c68FfA78ECFce1b1F7d9a
Deployer Address	0xBc631F653EB4a90CfDC7D7faEFE3d189949e36C8
Owner Address	0x32fE0ba05C9ef96A7C78671E524bEE7732fC744E
Tax Fees Buy	13%
Tax Fees Sell	13%
Gas Used for Buy	*will be update after dex listing*
Gas Used for Sell	*will be update after dex listing*
Contract Created	Sep-29-2022 11:28:38 AM +UTC
Initial Liquidity	*will be update after dex listing*
Liquidity Status	Locked
Unlocked Date	*will be update after dex listing*
Verified CA	Yes
Compiler	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Sol License	MIT License
Other	default evmVersion

TAX

BUY	13%	address	**SELL**	13%
Buy Fee	13%	send to RYS contract address 0xDe95A749Cbb93940A06c68FfA78ECFce1b1F7d9a	Sell Fee	13%

Token Holder



Txn Hash	Method ⓘ	Age	From		To	Quantity
0xaad6bd872e9dadb11d...	0x60806040	15 hrs 24 mins ago	Null Address: 0x000...000	→	0x32fe0ba05c9ef96a7c7...	20,000,000,000

[Download CSV Export ⭳]

Team Review

The RayonsEnergy team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 41 people in their telegram group (count in audit date).

Official Website And Social Media

Website: https://rayonsenergy.com/

Telegram Group: https://t.me/RayonsEnergy_En

Twitter: https://twitter.com/rayonsenergy

MANUAL CODE REVIEW

● Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked. Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problem.

● Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IBEP20

```
interface IBEP20 {
  /**
   * @dev Returns the number of tokens in existence.
   */
  function totalSupply() external view returns (uint256);
…
  function balanceOf(address account) external view returns (uint256);
…
  function transfer(address recipient, uint256 amount) external returns (bool);
…
  function allowance(address owner, address spender) external view returns (uint256);
…
  function approve(address spender, uint256 amount) external returns (bool);
…
  function transferFrom(
    address sender,
    address recipient,
    uint256 amount
  ) external returns (bool);

  /**
   * @dev Emitted when `value` tokens are moved from one account (`from`) to
   * another (`to`).
   *
   * Note that `value` may be zero.
   */
  event Transfer(address indexed from, address indexed to, uint256 value);
…
}
```

IBEP20 Normal Base Template

2. SafeMath Contract

```
library SafeMath {
...
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }
...
    function sub(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
  ...
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath contract

3. Rayons Energy Contract

```
contract RYSToken is IBEP20, Ownable {
    /*=== SafeMath ===*/
    using SafeMath for uint256;
    using Address for address;
    /*=== Endereços ===*/
    address private burnAddress = address(0); // Endereço de
Queima
    address private internalOperationAddress; // Distribui as
Recompensas para os Holders
    IUniswapV2Router02 public  uniswapV2Router; // Endereço Router
    address public  uniswapV2Pair; // Par LGK/BNB
    /*=== Mapeamento ===*/
    mapping (address => uint256) private _balance; // Saldo dos
Holders
    mapping (address => bool) public _excludeFromFee; // Nao paga
Taxas
    mapping (address => bool) private automatedMarketMakerPairs;
// Automatizado de Trocas
    mapping (address => mapping(address => uint256)) private
_allowances; // Subsidio
    mapping (address => bool) public isTimelockExempt; // Nao tem
Tempo de Espera
    mapping (address => uint) public cooldownTimerBuy; // Tempo de
Compra
    mapping (address => uint) public cooldownTimerSell; // Tempo
de Venda
    /*=== Unitarios ===*/
    uint8 private _decimals = 18;
    uint256 public cooldownTimerInterval; // Tempo de espera entre
compra e venda
    uint256 private _decimalFactor = 10**_decimals; // Fator
Decimal
    uint256 private _tSupply = 20000000000 * _decimalFactor; //
Supply Legitimik
    uint256 public buyFee = 13; // Taxa de Compra
    uint256 private previousBuyFee; // Armazena Taxa de Compra
    uint256 public sellFee = 13; // Taxa de Venda
    uint256 private previousSellFee; // Armazena Taxa de Venda
    uint256 private limitBurn = 200000000 * _decimalFactor;
    uint256 public liquidityPercent = 20; // Taxa de Liquidez 20%
    uint256 public maxWalletBalance = 5000000000 * _decimalFactor;
// 1% do Supply
    uint256 public maxBuyAmount = 5000000000 * _decimalFactor; //
1% do Supply
    uint256 public maxSellAmount = 5000000000 * _decimalFactor; //
1% do Supply
    uint256 public numberOfTokensToSwapToLiquidity = 500 *
_decimalFactor; // 0,05% do Supply
    /*=== Boolean ===*/
    bool private inSwapAndLiquify;
    bool private coolDownUser;
    bool private swapping;
    bool private activeDividends = true;
    bool public buyCooldownEnabled = true;
    bool public sellCooldownEnabled = true;
    bool private isSendToken = true;
    bool private blackEnabled = true;
    bool private swapAndLiquifyEnabled = true;
    /*=== Strings ===*/
    string private _name = "RYRS";
    string private _symbol =  "$RYS";
    /*=== Modifiers ===*/
    modifier lockTheSwap {
        inSwapAndLiquify = true;
        _;
        inSwapAndLiquify = false;
    }
    modifier lockCoolDown {
        coolDownUser = true;
        _;
        coolDownUser = false;
    }
    /*=== Construtor ===*/
    constructor() {
        //  IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0xD99D1c33F9fC3444f8101754aBC46c52416550D1); //
PancakeSwap Router Testnet
        IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); //
PancakeSwap Router Mainnet
        address pairCreated =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(t
his), _uniswapV2Router.WETH()); // Gera o Par RYS/BNB
        uniswapV2Router = _uniswapV2Router; // Armazena Rota
        uniswapV2Pair = pairCreated; // Armazena Par
        cooldownTimerInterval = 5;
        _balance[0x32fE0ba05C9ef96A7C78671E524bEE7732fC744E] =
_tSupply; // Define Owner como Detentor dos _Tokens
        internalOperationAddress = owner(); // Define Endereço de
Operações

_excludeFromFee[0x32fE0ba05C9ef96A7C78671E524bEE7732fC744E] =
true; // Define Owner como True para não pagar Taxas
        _excludeFromFee[owner()] = true; // Define Owner como True
para não pagar Taxas
        _excludeFromFee[address(this)] = true; // Define Contrato
como True para não pagar Taxas
        _excludeFromFee[internalOperationAddress] = true; //
Define internalOperationAddress como True para não pagar Taxas

isTimelockExempt[0x32fE0ba05C9ef96A7C78671E524bEE7732fC744E] =
true; // Owner Nao tem tempo de espera para compra e venda
        isTimelockExempt[owner()] = true; // Owner Nao tem tempo
de espera para compra e venda
        isTimelockExempt[address(this)] = true; // Contrato Nao
tem tempo de espera para compra e venda
       _setAutomatedMarketMakerPair(pairCreated, true); // Pair é
o Automatizador de Transações
       _approve(owner(), address(uniswapV2Router), ~uint256(0));
// Aprova Tokens para Add Liquidez
        emit Transfer(address(0),
0x32fE0ba05C9ef96A7C78671E524bEE7732fC744E, _tSupply); // Emite um
Evento de Cunhagem
    }
    /*=== Receive ===*/
    receive() external payable {}
    /*=== Public View ===*/
    function name() public view override returns(string memory) {
return _name; } // Nome do Token
    function symbol() public view override returns(string memory)
{ return _symbol; } // Simbolo do Token
    function decimals() public view override returns(uint8) {
return _decimals; } // Decimais
```

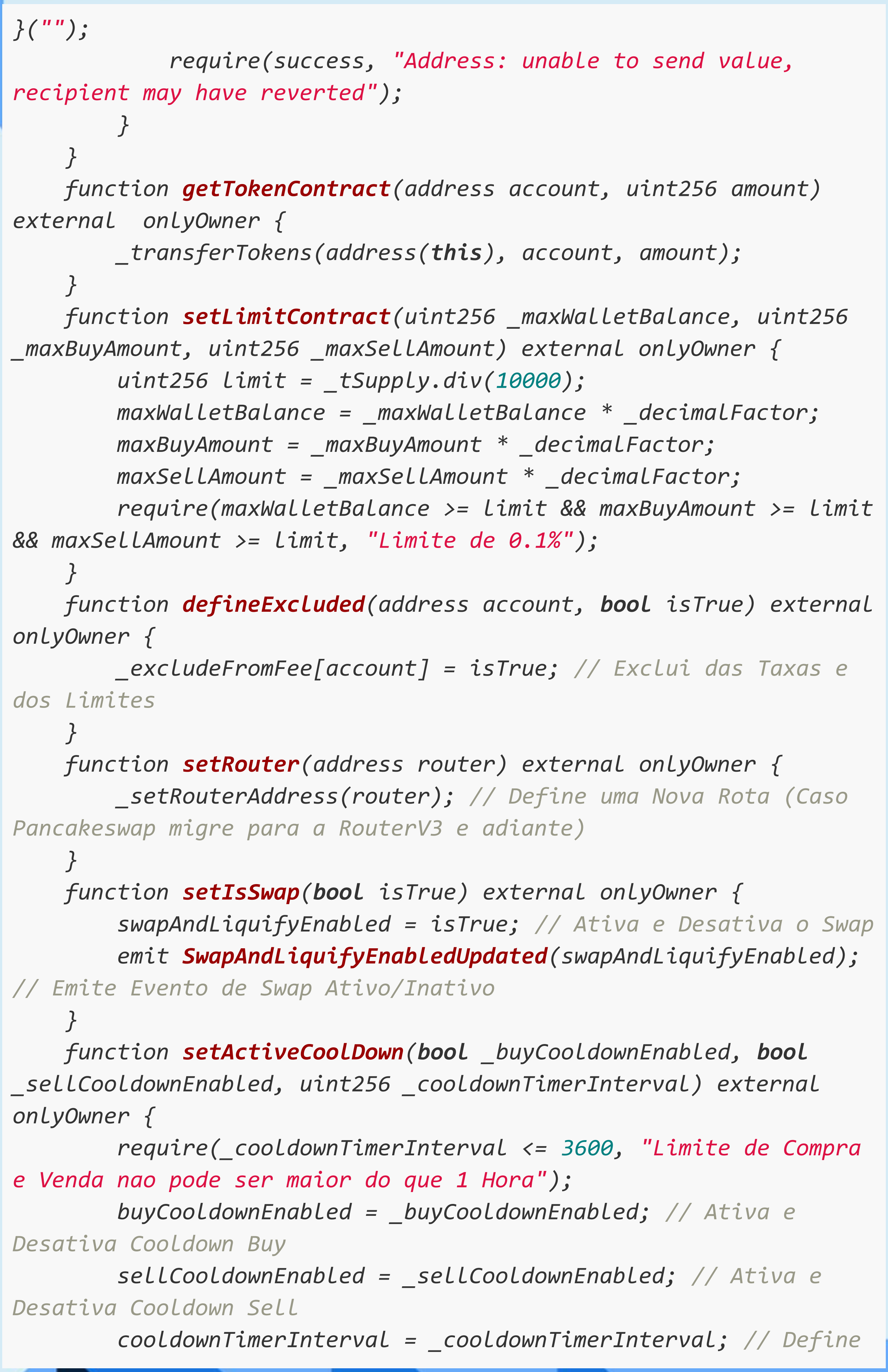
```
    function totalSupply() public view override returns(uint256) {
return _tSupply; } // Supply Total
    function balanceOf(address account) public view override
returns(uint256) { return _balance[account]; } // Retorna o Saldo
em Carteira
    function allowance(address owner, address spender) public view
override returns(uint256) { return _allowances[owner][spender]; }
// Subsidio Restante
        /*=== Eventos ===*/
    event SetAutomatedMarketMakerPair(address indexed pair, bool
indexed value);
    event SwapAndLiquify(uint256 tokensSwapped, uint256
ethReceived, uint256 tokensIntoLiquidity);
    event SwapAndLiquifyEnabledUpdated(bool indexed enabled);
    event LiquidityAdded(uint256 tokenAmountSent, uint256
ethAmountSent, uint256 liquidity);
    event SentBNBInternalOperation(address usr, uint256 amount);
    /*=== Private/Internal ===*/
    function _setRouterAddress(address router) private {
        IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(router); // Router
        address pairCreated =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(t
his), _uniswapV2Router.WETH()); // Gera o Par LGK/BNB
        uniswapV2Router = _uniswapV2Router; // Armazena Rota
        uniswapV2Pair = pairCreated; // Armazena Par
        _setAutomatedMarketMakerPair(uniswapV2Pair, true); //
Armazena o novo Par como o Automatizador de Trocas
    }
    function _setAutomatedMarketMakerPair(address pair, bool
value) private {
        require(automatedMarketMakerPairs[pair] != value, "O pair
de AutomatedMarketMakerPair ja esta definido para esse valor");
        automatedMarketMakerPairs[pair] = value; // Booleano
        emit SetAutomatedMarketMakerPair(pair, value); // Emite um
Evento para um Novo Automatizador de Trocas
    }
    function _approve(address owner, address spender, uint256
amount) internal {
        require(owner != address(0), "Owner nao pode ser Address
0");
        require(spender != address(0), "Owner nao pode ser Address
0");
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
    function _spendAllowance(address owner, address spender,
uint256 amount) internal {
        uint256 currentAllowance = allowance(owner, spender);
        if(currentAllowance != type(uint256).max) {
            require(currentAllowance >= amount, "subsidio
insuficiente");
            _approve(owner, spender, currentAllowance - amount);
        }
    }
    function _unlimitedAddress(address account) internal view
returns(bool) {
        if(_excludeFromFee[account]) {
            return true;
        }
        else {return false;}
    }
    function buyCoolDown(address to) private lockCoolDown {
        cooldownTimerBuy[to] = block.timestamp; // Ativa o Tempo
de Compra
    }
    function sellCoolDown(address from) private lockCoolDown  {
        cooldownTimerSell[from] = block.timestamp; // Ativa o
Tempo de Venda
    }
    function lockToBuyOrSellForTime(uint256 lastBuyOrSellTime,
uint256 lockTime) private lockCoolDown returns (bool) {
        uint256 crashTime = lastBuyOrSellTime + lockTime;
        uint256 currentTime = block.timestamp;
        if(currentTime >= crashTime) {
            return true;
        }

        return false;
    }
    function getFromLastPurchaseBuy(address walletBuy) private
view returns (uint) {
        return cooldownTimerBuy[walletBuy];
    }
    function getFromLastSell(address walletSell) private view
returns (uint) {
        return cooldownTimerSell[walletSell];
    }
    function _beforeTokenTransfer( address from, address to,
uint256 amount ) internal virtual{}
    function _afterTokenTransfer( address from, address to,
uint256 amount ) internal virtual{}
    function _transferTokens(address from, address to, uint256
amount) internal {
        require(to != from, "Nao pode enviar para o mesmo
Endereco");
        require(amount > 0, "Saldo precisa ser maior do que
Zero");
        _beforeTokenTransfer(from, to, amount);



        uint256 fromBalance = _balance[from];
        require(fromBalance >= amount, "Voce nao tem Limite de
Saldo");
        _balance[from] = fromBalance - amount;

        uint256 contractTokenBalance = balanceOf(address(this));
        if (!automatedMarketMakerPairs[from] &&
automatedMarketMakerPairs[to]) {
            swapping = true;
            liquify( contractTokenBalance, from );
            swapping = false;
        }

        bool takeFee = true;

        if(_excludeFromFee[from] || _excludeFromFee[to]){
            takeFee = false;
        }

        if(!takeFee) removeAllFee(); // Remove todas as Taxa

            uint256 fees; // Taxas de Compra, Vendas e
Transferencias!
```

```
            if(automatedMarketMakerPairs[from]) {
                fees = amount.mul(buyFee).div(100); // Define taxa
de Compra
                if (amount > maxBuyAmount &&
!_unlimitedAddress(from) && !_unlimitedAddress(to)) {
                    revert("Montante de Venda nao pode ultrapassar
limite");
                }

                if(buyCooldownEnabled && !isTimelockExempt[to] &&
!coolDownUser) {

require(lockToBuyOrSellForTime(getFromLastPurchaseBuy(to),
cooldownTimerInterval), "Por favor, aguarde o cooldown entre as
compras");
                    buyCoolDown(to);
                }
            }
            else if(automatedMarketMakerPairs[to]) {
                fees = amount.mul(sellFee).div(100); // Define
taxa de Venda
                if (amount > maxSellAmount &&
!_unlimitedAddress(from) && !_unlimitedAddress(to)) {
                    revert("Montante de Venda nao pode ultrapassar
limite");
                }

                if(sellCooldownEnabled && !isTimelockExempt[from]
&& !coolDownUser) {

require(lockToBuyOrSellForTime(getFromLastSell(from),
cooldownTimerInterval), "Por favor, aguarde o cooldown entre as
vendas");
                    sellCoolDown(from);
                }
            }

            if(maxWalletBalance > 0 && !_unlimitedAddress(from) &&
!_unlimitedAddress(to) && !automatedMarketMakerPairs[to]) {
                uint256 recipientBalance = balanceOf(to); //
Define o Maximo por Wallet
```

```
                require(recipientBalance.add(amount) <=
maxWalletBalance, "Nao pode Ultrapassar o limite por Wallet");
            }

            if(fees != 0) {
                amount = amount.sub(fees);
                _balance[address(this)] += fees;
                emit Transfer(from, address(this), fees); // Emite
um Evento de Envio de Taxas
            }

            _balance[to] += amount;




            emit Transfer(from, to, amount); // Emite um Evento de
Transferencia
            _afterTokenTransfer(from, to, amount);
        if(!takeFee) restoreAllFee(); // Retorna todas as Taxa
    }
    function removeAllFee() private {
        previousBuyFee = buyFee; // Armazena Taxa Anterior
        previousSellFee = sellFee; // Armazena Taxa Anterior
        buyFee = 0; // Taxa 0
        sellFee = 0; // Taxa 0
    }
    function restoreAllFee() private {
        buyFee = previousBuyFee; // Restaura Taxas
        sellFee = previousSellFee; // Restaura Taxas
    }
    function liquify(uint256 contractTokenBalance, address sender)
internal {

        if (contractTokenBalance >=
numberOfTokensToSwapToLiquidity) contractTokenBalance =
numberOfTokensToSwapToLiquidity; // Define se a Quantidade de
Tokens para

        bool isOverRequiredTokenBalance = ( contractTokenBalance
>= numberOfTokensToSwapToLiquidity ); // Booleano
```

```
        if ( isOverRequiredTokenBalance && swapAndLiquifyEnabled
&& !inSwapAndLiquify && (!automatedMarketMakerPairs[sender]) ) {
            uint256 tokenLiquidity =
contractTokenBalance.mul(liquidityPercent).div(100); // Quantidade
de Tokens que vai para Liquidez
            uint256 toSwapBNB =
contractTokenBalance.sub(tokenLiquidity); // Quantidade de Tokens
para Venda
            _swapAndLiquify(tokenLiquidity); // Adiciona Liquidez
            _sendBNBToContract(toSwapBNB); // Troca Tokens por BNB
        }

    }
    function _swapAndLiquify(uint256 amount) private lockTheSwap {
        uint256 half = amount.div(2); // Divide para Adicionar
Liquidez
        uint256 otherHalf = amount.sub(half); // Divide para
Adicionar Liquidez
        uint256 initialBalance = address(this).balance; //
Armazena o Saldo Inicial em BNB
        _swapTokensForEth(half); // Efetua a troca de Token por
BNB
        uint256 newBalance =
address(this).balance.sub(initialBalance); // Saldo atual em BNB -
Saldo Antigo
        _addLiquidity(otherHalf, newBalance); // Adiciona Liquidez
        emit SwapAndLiquify(half, newBalance, otherHalf); // Emite
Evento de Swap
    }
    function _sendBNBToContract(uint256 tAmount) private
lockTheSwap {
         _swapTokensForEth(tAmount); // Vende os Tokens por BNB e
envia para o Contrato
        if(isSendToken) {
            uint256 initialBalance = address(this).balance;
            if(initialBalance > 0) {

                (bool sent, ) =
internalOperationAddress.call{value: address(this).balance}("");
                if(sent) {
                    emit
SentBNBInternalOperation(internalOperationAddress,
initialBalance);
                }
            }
        }
    }

    function _swapTokensForEth(uint256 tokenAmount) private {
        address[] memory path = new address[](2); // Path Memory
para inicia a venda dos Tokens
        path[0] = address(this); // Endereço do Contrato
        path[1] = uniswapV2Router.WETH(); // Par de Troca (BNB)
        _approve(address(this), address(uniswapV2Router),
tokenAmount); // Aprova os Tokens para Troca

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
(
            tokenAmount, // Saldo para Swap
            0, // Amount BNB
            path, // Path [address(this), uniswapV2Router.WETH()]
            address(this), // Endereço de Taxa
            block.timestamp // Timestamp
        );
    }
    function _addLiquidity(uint256 tokenAmount, uint256 ethAmount)
private {
        _approve(address(this), address(uniswapV2Router),
tokenAmount);
        (uint256 tokenAmountSent, uint256 ethAmountSent, uint256
liquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount, // Saldo para Liquidez
            0, // Slippage 0
            0, // Slippage 0
            owner(), // Owner Adiciona Liquidez
            block.timestamp // Timestamp
        );
        emit LiquidityAdded(tokenAmountSent, ethAmountSent,
liquidity); // Emite Evento de Liquidez
    }

    /*=== Public/External ===*/
    function approve(address spender, uint256 amount) public
```

```
override returns(bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }
    function transfer(address to, uint256 amount) public override
returns(bool){
        _transferTokens(_msgSender(), to, amount);
        return true;
    }
    function transferFrom(address from, address to, uint256
amount) public override returns(bool) {
        _spendAllowance(from, _msgSender(), amount);
         _transferTokens(from, to, amount);
        return true;
    }

    /*=== Funções Administrativas ===*/

    function changeAutomatedMarketMakerPair(address pair, bool
value) external onlyOwner {
        require(pair != uniswapV2Pair, "uniswapV2Pair nao pode ser
removido de AutomatedMarketMakerPair");
        _setAutomatedMarketMakerPair(pair, value); // Define um
Novo Automatizador de Trocas
    }
    function changeFees(uint256 _buyFee, uint256 _sellFee, uint256
_liquidityPercent) external onlyOwner {
        require(_buyFee <= 25 && _sellFee <= 25, "A taxa nao pode
ser maior do que 25%");
        buyFee = _buyFee;
        sellFee = _sellFee;
        liquidityPercent = _liquidityPercent;
    }
    function changeAddress(address _internalOperationAddress)
external onlyOwner {
        internalOperationAddress = _internalOperationAddress; //
Define Endereço de Operações
    }
    function removeBNB() external payable onlyOwner {
        uint256 balance = address(this).balance;
        if(balance > 0) {
            (bool success, ) = _msgSender().call{ value: balance
```



```
}("");
            require(success, "Address: unable to send value, 
recipient may have reverted");
        }
    }
    function getTokenContract(address account, uint256 amount) 
external  onlyOwner {
        _transferTokens(address(this), account, amount);
    }
    function setLimitContract(uint256 _maxWalletBalance, uint256 
_maxBuyAmount, uint256 _maxSellAmount) external onlyOwner {
        uint256 limit = _tSupply.div(10000);
        maxWalletBalance = _maxWalletBalance * _decimalFactor;
        maxBuyAmount = _maxBuyAmount * _decimalFactor;
        maxSellAmount = _maxSellAmount * _decimalFactor;
        require(maxWalletBalance >= limit && maxBuyAmount >= limit 
&& maxSellAmount >= limit, "Limite de 0.1%");
    }
    function defineExcluded(address account, bool isTrue) external 
onlyOwner {
        _excludeFromFee[account] = isTrue; // Exclui das Taxas e 
dos Limites
    }
    function setRouter(address router) external onlyOwner {
        _setRouterAddress(router); // Define uma Nova Rota (Caso 
Pancakeswap migre para a RouterV3 e adiante)
    }
    function setIsSwap(bool isTrue) external onlyOwner {
        swapAndLiquifyEnabled = isTrue; // Ativa e Desativa o Swap
        emit SwapAndLiquifyEnabledUpdated(swapAndLiquifyEnabled); 
// Emite Evento de Swap Ativo/Inativo
    }
    function setActiveCoolDown(bool _buyCooldownEnabled, bool 
_sellCooldownEnabled, uint256 _cooldownTimerInterval) external 
onlyOwner {
        require(_cooldownTimerInterval <= 3600, "Limite de Compra 
e Venda nao pode ser maior do que 1 Hora");
        buyCooldownEnabled = _buyCooldownEnabled; // Ativa e 
Desativa Cooldown Buy
        sellCooldownEnabled = _sellCooldownEnabled; // Ativa e 
Desativa Cooldown Sell
        cooldownTimerInterval = _cooldownTimerInterval; // Define
```

```
Segundos entre Compra e Venda
    }
    function activeSendDividends(bool _isSendToken) external
onlyOwner {
        isSendToken = _isSendToken;
    }
    function setBurn(uint256 _limitBurn) external onlyOwner {
        limitBurn = _limitBurn * _decimalFactor;
    }
    function burn(uint256 bAmount) external onlyOwner {
        require(bAmount <= limitBurn, "Nao pode queimar mais do
que o programado");
        _tSupply -= bAmount;
         _balance[_msgSender()] -= bAmount;
        emit Transfer(_msgSender(), burnAddress, bAmount);
    }
    function setSwapAmount(uint256 tAmount) external onlyOwner {
        numberOfTokensToSwapToLiquidity = tAmount *
_decimalFactor; // Define a quantidade de Tokens que o Contrato
vai Vender
    }



}
```

READ CONTRACT (ONLY NEED TO KNOW)

1. buyCooldownEnabled
true bool
(Shows Contract buy cooldown status)

2. buyFee
13 uint256
(Shows buy fee)

3. sellFee
13 uint256
(Shows sell fee)

4. name
RYS string
(Function for read Token name)

WRITE CONTRACT

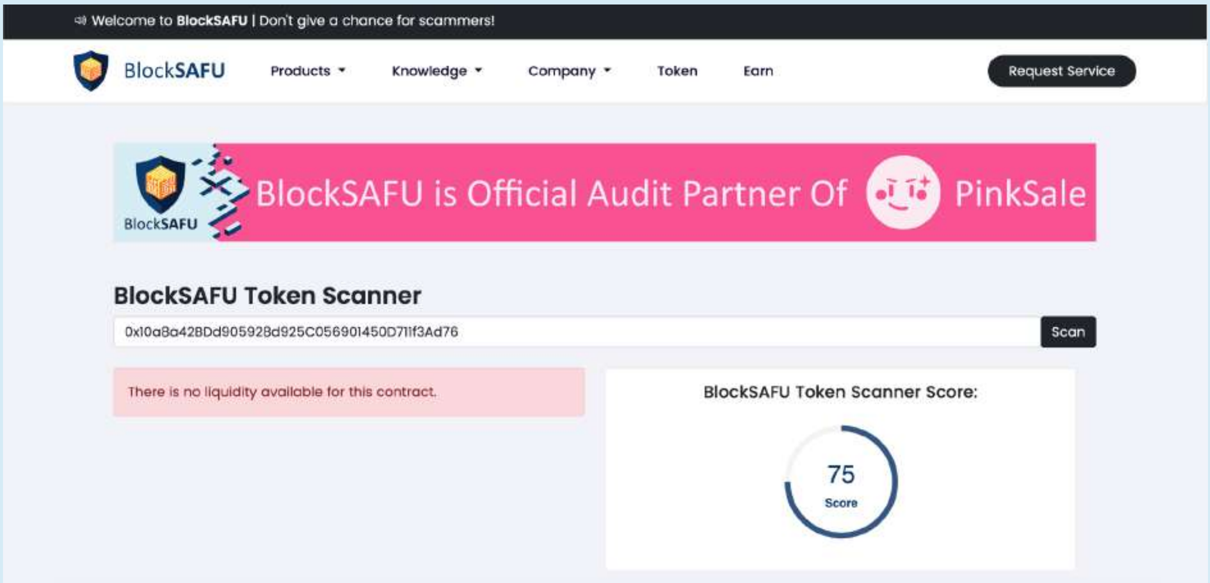
1. renounceOwnership
(Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner)

2. transferOwnership
newOwner (address)
(Its function is to change the owner)

3. changeBlacklistUser
user (address)
isTrue (bool)
(its function to blacklist address)

BlockSAFU TOKEN SCANNER

https://blocksafu.com/token-scanner


Welcome to BlockSAFU | Don't give a chance for scammers!
BlockSAFU
Products
Knowledge
Company
Token
Earn
Request Service
BlockSAFU
BlockSAFU is Official Audit Partner Of PinkSale
Scan

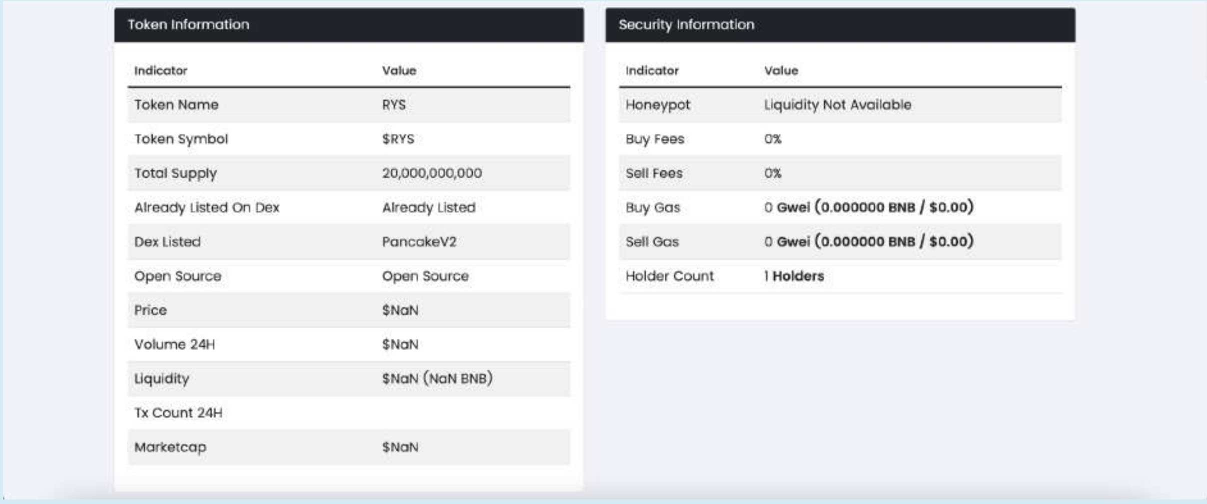
BlockSAFU Token Scanner

0x10a8a42BDd905928d925C056901450D711f3Ad76

There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

75
Score

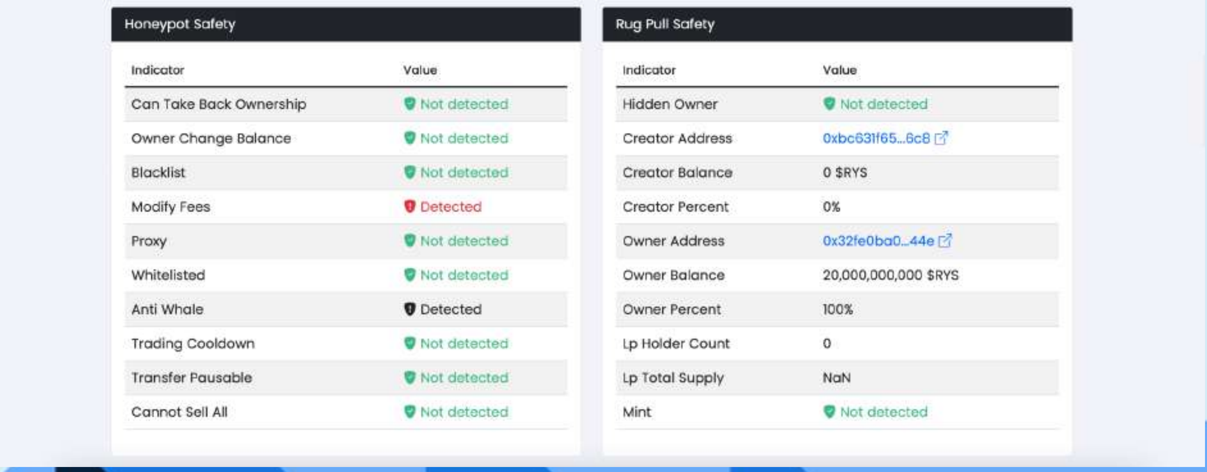


Token Information

Indicator	Value
Token Name	RYS
Token Symbol	$RYS
Total Supply	20,000,000,000
Already Listed On Dex	Already Listed
Dex Listed	PancakeV2
Open Source	Open Source
Price	$NaN
Volume 24H	$NaN
Liquidity	$NaN (NaN BNB)
Tx Count 24H	
Marketcap	$NaN

Security Information

Indicator	Value
Honeypot	Liquidity Not Available
Buy Fees	0%
Sell Fees	0%
Buy Gas	0 **Gwei (0.000000 BNB / $0.00)**
Sell Gas	0 **Gwei (0.000000 BNB / $0.00)**
Holder Count	1 **Holders**



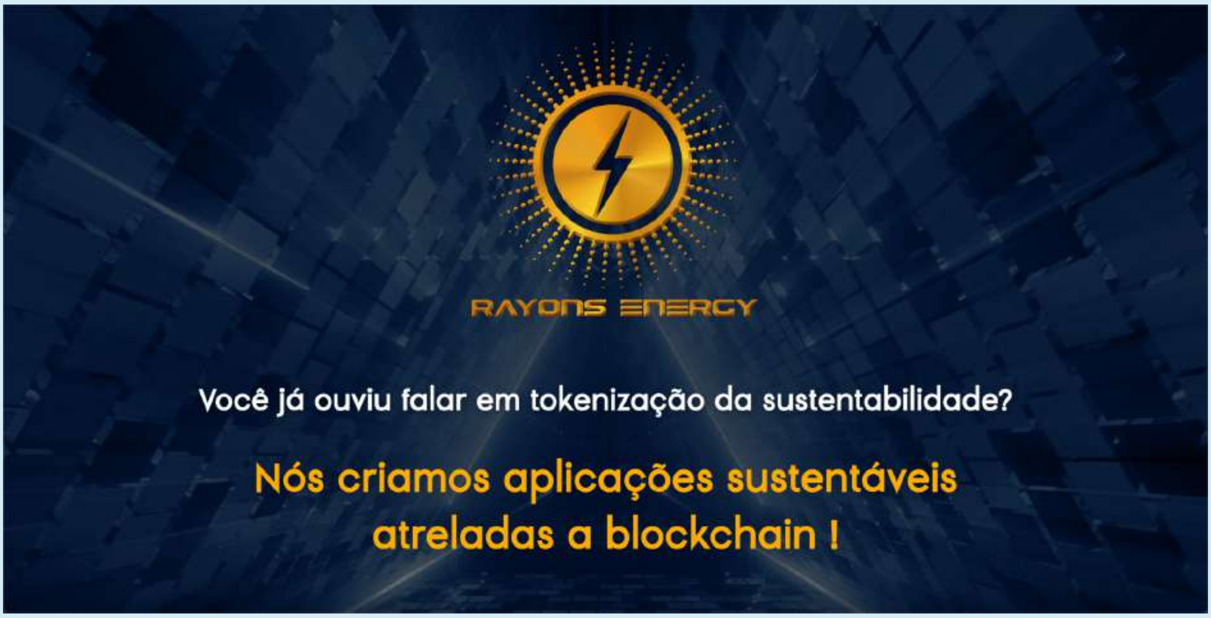
Honeypot Safety

Indicator	Value
Can Take Back Ownership	✓ Not detected
Owner Change Balance	✓ Not detected
Blacklist	✓ Not detected
Modify Fees	! Detected
Proxy	✓ Not detected
Whitelisted	✓ Not detected
Anti Whale	! Detected
Trading Cooldown	✓ Not detected
Transfer Pausable	✓ Not detected
Cannot Sell All	✓ Not detected

Rug Pull Safety

Indicator	Value
Hidden Owner	✓ Not detected
Creator Address	0xbc631f65...6c8
Creator Balance	0 $RYS
Creator Percent	0%
Owner Address	0x32fe0ba0...44e
Owner Balance	20,000,000,000 $RYS
Owner Percent	100%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	✓ Not detected

WEBSITE REVIEW



RAYONS ENERGY

Você já ouviu falar em tokenização da sustentabilidade?

Nós criamos aplicações sustentáveis atreladas a blockchain !

- Mobile Friendly
- Contains no code error
- SSL Secured (By Let’s Encrypt SSL)

Web-Tech stack: Nicepages, LazySizes

Domain .com (enom) - Tracked by whois

First Contentful Paint:	779s
Fully Loaded Time	2.0s
Performance	81%
Accessibility	85%
Best Practices	100%
SEO	80%

RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

- 🟢 Locked Liquidity (Locked by pinksale)

  will be updated after listing dex

- 🟢 TOP 5 Holder.

  will be updated after listing dex

- 🟢 The Team KYC by PinkSale

HONEYPOT REVIEW

- 🟢 Ability to sell.
- 🟢 The owner can’t set fees over 25%
- 🟢 The owner can’t set max tx amount below 0.1% of total supply

Note: Please check the disclaimer above and note, that the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.