



BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT



Project: Cristomonedas

Website: <https://cristomonedas.com/en/>



BlockSAFU Score:

66

Contract Address:

0xe748d5C3A572f14D07D4a16E0ef38A96ff891156

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

DISCLAIMER

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. BlockSAFU's 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 11% (owner can't set fees over 25%).
- Sell 11% (owner can't set fees over 25%).

Tx Amount

- Owner cannot set max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

SMART CONTRACT REVIEW

Token Name	CRISTOMONEDAS
Token Symbol	CSM
Token Decimal	6
Total Supply	7,000,000 CSM
Contract Address	0xe748d5C3A572f14D07D4a16E0ef38A96ff891156
Deployer Address	0xeB920D80799bbAFe08e2881AB8BB87C865e450Fc
Owner Address	0x705Efc578d6B19fEb91EB826a6BFDB7985e17B37
Tax Fees Buy	6%
Tax Fees Sell	6%
Gas Used for Buy	will be update after token listing on dex
Gas Used for Sell	will be update after token listing on dex
Contract Created	Sep-22-2022 08:20:40 AM +UTC
Initial Liquidity	will be update after token listing on dex
Liquidity Status	Locked
Unlocked Date	will be update after token listing on dex
Verified CA	Yes
Compiler	v0.8.16+commit.07a7930e
Optimization	Yes with 200 runs
Sol License	MIT License
Other	default evmVersion

TAX

BUY	6%	address	SELL	6%
BNB Fee	2%	For Marketing	BNB Fee	2%
Burn Fee	2%	For Burn	Burn fee	2%
Contract Fee	0%	0xde491C65E507d281B6a3688d11e8fC222eee0975	Contract Fee	0%
Liquidity fee	1%	automatic add liquidity	Liquidity Fee	1%
Reflection fee	0%	For Reflection	Reflection Fee	0%
Token fee	1%	0xbad39e4a73cb1b6a669467e81954ffbfcb70c5b	Token fee	1%

Token Holder

Rank	Address	Quantity	Percentage	Analytics
1	0x705efc578d6b19feb91eb826a6bfdb7985e17b37	7,000,000	100.0000% <div><div></div></div>	Analytics

[Download CSV Export [↗](#)]

Team Review

The CSM team has a nice website, their website is professionally built and the Smart contract is well developed

Official Website And Social Media

Website: <https://cristomonedas.com/en/>

Facebook:

<https://www.facebook.com/Cristomonedas-103325522545623>

MANUAL CODE REVIEW

● Minor-risk

2 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);
```

2. Owner can set calmdown trading but **no limit** time

```
function Launch_Settings_02__Set_Launch_Limits(  
  
    uint256 Launch_Buy_Delay_Seconds,  
    uint256 Launch_Transaction_Limit_TOKENS,  
    uint256 Launch_Phase_Length_Minutes  
  
    ) external onlyOwner {  
  
    max_Tran_Launch  = Launch_Transaction_Limit_TOKENS * 10 **  
_decimals;  
    Launch_Buy_Delay = Launch_Buy_Delay_Seconds;  
    Launch_Length    = Launch_Phase_Length_Minutes * 60;  
    }  
}
```

● Medium-risk

1 medium-risk code issues found

Should be fixed, could bring problems.

```
function Launch_Settings_01__Blacklist_Bots(
    address Wallet,
    bool true_or_false

) external onlyOwner {

    // Buyer Protection - Blacklisting can only be done before
launch
    if (true_or_false){require(LaunchTime == 0, "E08");}
    _isBlacklisted[Wallet] = true_or_false;
}
```

1. Owner can set blacklist address

Recommendation: remove this function

● 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. IERC20

```
interface IERC20 {  
    /**  
     * @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);  
    ...  
}
```

IERC20 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. CSM Contract

```
contract Contract is Context, IERC20 {

    using SafeMath for uint256;
    using Address for address;

    // Contract Wallets
    address private _owner; //
    Contract Owner
    address public Wallet_Liquidity; // LP
    Token Collection Wallet for Auto LP
    address public Wallet_Tokens; // Token
    Fee Collection Wallet
    address payable public Wallet_BNB; // BNB Fee
    Collection Wallet
    address payable public Wallet_TBG_AFF; //
    TokensByGEN Affiliate Wallet and Discount Code

    // Contract fee (1% ongoing if applicable) is sent to fee
    collection contract
    address payable public constant feeCollector =
    payable(0xde491C65E507d281B6a3688d11e8fC222eee0975);

    // Token Info
    string private _name;
    string private _symbol;
    uint256 private _decimals;
    uint256 private _tTotal;

    // Token social links will appear on BSCScan
    string private _Website;
    string private _Telegram;
    string private _LP_Locker_URL;

    // Wallet and transaction Limits
    uint256 private max_Hold;
    uint256 private max_Trans;

    // Fees - Set fees before opening trade
    uint256 public _Fee__Buy_Burn;
    uint256 public _Fee__Buy_Contract;
    uint256 public _Fee__Buy_Liquidity;
```

```

uint256 public _Fee__Buy_BNB;
uint256 public _Fee__Buy_Reflection;
uint256 public _Fee__Buy_Tokens;

uint256 public _Fee__Sell_Burn;
uint256 public _Fee__Sell_Contract;
uint256 public _Fee__Sell_Liquidity;
uint256 public _Fee__Sell_BNB;
uint256 public _Fee__Sell_Reflection;
uint256 public _Fee__Sell_Tokens;

// Upper limit for fee processing trigger
uint256 private swap_Max;

// Total fees that are processed on buys and sells for swap
and liquify calculations
uint256 private _SwapFeeTotal_Buy;
uint256 private _SwapFeeTotal_Sell;

// Track contract fee
uint256 private ContractFee;

// Supply Tracking for RFI
uint256 private _rTotal;
uint256 private _tFeeTotal;
uint256 private constant MAX = ~uint256(0);

// Launch Phase Settings
uint256 private max_Tran_Launch;
uint256 private Launch_Buy_Delay;
uint256 private Launch_Length;

// Affiliate Tracking
IERC20 GEN =
IERC20(0x7d7a7f452e04C2a5df792645e8bfaF529aDcCEcf); // GEN - For
tracking affiliate level
IERC20 AFT =
IERC20(0x98A70E83A53544368D72940467b8bB05267632f4); // TokensByGEN
Affiliate Tracker Token

uint256 private constant Tier_2 = 500000 * 10**9;

```

```

uint256 private constant Tier_3 = 1000000 * 10**9;

// Set factory
IUniswapV2Router02 public uniswapV2Router;
address public uniswapV2Pair;

constructor (string memory _TokenName,
            string memory _TokenSymbol,
            uint256 _TotalSupply,
            uint256 _Decimals,
            address payable _OwnerWallet,
            address payable _DiscountCode,
            uint256 _ContractFee) {

    // Set owner
    _owner = _OwnerWallet;

    // Set basic token details
    _name = _TokenName;
    _symbol = _TokenSymbol;
    _decimals = _Decimals;
    _tTotal = _TotalSupply * 10**_decimals;
    _rTotal = (MAX - (MAX % _tTotal));

    // Wallet Limits - Set limits after deploying
    max_Hold = _tTotal;
    max_Tran = _tTotal;

    // Contract sell limit when processing fees
    swap_Max = _tTotal / 200;

    // Launch Phase control
    max_Tran_Launch = _tTotal;
    Launch_Buy_Delay = 0;
    Launch_Length = 5 * 60;

    // Set BNB, tokens, and liquidity collection wallets to owner
    (can be updated later)
    Wallet_BNB = payable(_OwnerWallet);
    Wallet_Liquidity = _OwnerWallet;
    Wallet_Tokens = _OwnerWallet;

```

```
// Set contract fee
ContractFee          = _ContractFee;

// Transfer token supply to owner wallet
_rOwned[_owner]      = _rTotal;

// Set TokensByGEN affiliate from Discount Code
Wallet_TBG_AFF       = payable(_DiscountCode);

// Set PancakeSwap Router Address
IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);

// Create initial liquidity pair with BNB on PancakeSwap
factory
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(t
his), _uniswapV2Router.WETH());
    uniswapV2Router = _uniswapV2Router;

// Wallets that are excluded from holding limits
_isLimitExempt[_owner] = true;
_isLimitExempt[address(this)] = true;
_isLimitExempt[Wallet_Burn] = true;
_isLimitExempt[uniswapV2Pair] = true;
_isLimitExempt[Wallet_Tokens] = true;

// Wallets that are excluded from fees
_isExcludedFromFee[_owner] = true;
_isExcludedFromFee[address(this)] = true;
_isExcludedFromFee[Wallet_Burn] = true;

// Set the initial liquidity pair
_isPair[uniswapV2Pair] = true;

// Exclude from Rewards
_isExcluded[Wallet_Burn] = true;
_isExcluded[uniswapV2Pair] = true;
_isExcluded[address(this)] = true;

// Push excluded wallets to array
```

```

    _excluded.push(Wallet_Burn);
    _excluded.push(uniswapV2Pair);
    _excluded.push(address(this));

    // Wallets granted access before trade is open
    _isWhiteListed[_owner] = true;

    // Emit Supply Transfer to Owner
    emit Transfer(address(0), _owner, _tTotal);

    // Emit ownership transfer
    emit OwnershipTransferred(address(0), _owner);

}

// Events
event OwnershipTransferred(address indexed previousOwner,
address indexed newOwner);
event updated_Wallet_Limits(uint256 max_Tran, uint256
max_Hold);
event updated_Buy_fees(uint256 Marketing, uint256 Liquidity,
uint256 Reflection, uint256 Burn, uint256 Tokens, uint256
Contract_Development_Fee);
event updated_Sell_fees(uint256 Marketing, uint256 Liquidity,
uint256 Reflection, uint256 Burn, uint256 Tokens, uint256
Contract_Development_Fee);
event updated_SwapAndLiquify_Enabled(bool
Swap_and_Liquify_Enabled);
event updated_trade_Open(bool TradeOpen);
event SwapAndLiquify(uint256 tokensSwapped, uint256
ethReceived, uint256 tokensIntoLiquidity);
event set_Contract_Fee(uint256 Contract_Development_Buy_Fee,
uint256 Contract_Development_Sell_Fee);

// Restrict function to contract owner only
modifier onlyOwner() {
    require(owner() == _msgSender(), "Ownable: caller is not
the owner");
    _;
}

```



```

uint256
Fee_When_Buying,
uint256
Fee_When_Selling,
string
memory Website,
string
memory Telegram,
string
memory Liquidity_Lock_URL,
string
memory Contract_Created_By) {

```

```

    string memory Creator = "https://tokensbygen.com/";

```

```

    uint256 Total_buy =  _Fee__Buy_Burn          +
                        _Fee__Buy_Contract      +
                        _Fee__Buy_Liquidity      +
                        _Fee__Buy_BNB           +
                        _Fee__Buy_Reflection    +
                        _Fee__Buy_Tokens        ;

```

```

    uint256 Total_sell =  _Fee__Sell_Burn        +
                        _Fee__Sell_Contract      +
                        _Fee__Sell_Liquidity      +
                        _Fee__Sell_BNB           +
                        _Fee__Sell_Reflection    +
                        _Fee__Sell_Tokens        ;

```

```

    uint256 TranLimit = max_Tran / 10 ** _decimals;

```

```

    if (LaunchPhase && (max_Tran_Launch < max_Tran)){
        TranLimit = max_Tran_Launch / 10 ** _decimals;
    }

```

```

    // Return Token Data
    return (_name,
            _symbol,

```

```

        _decimals,
        _owner,
        TranLimit,
        max_Hold / 10 ** _decimals,
        Total_buy,
        Total_sell,
        _Website,
        _Telegram,
        _LP_Locker_URL,
        Creator);

}

// Burn (dead) address
address public constant Wallet_Burn =
0x0000000000000000000000000000000000000000000000000000000000000000dEaD;

// Swap triggers
uint256 private swapTrigger = 11;
uint256 private swapCounter = 1;

// SwapAndLiquify - Automatically processing fees and adding
Liquidity
bool public inSwapAndLiquify;
bool public swapAndLiquifyEnabled;

// Launch settings
bool public TradeOpen;
bool private LaunchPhase;
uint256 private LaunchTime;

// No fee on wallet-to-wallet transfers
bool noFeeW2W = true;

// Deflationary Burn - Tokens Sent to Burn are removed from
total supply if set to true
bool public deflationaryBurn;

// Take fee tracker
bool private takeFee;

```

/*

CONTRACT SET UP AND DEPLOYMENT GUIDE

* /

/ *

DECIDE IF BURN WALLET WILL BE DEFLATIONARY

By default this is set to false

If you change this to true, when tokens are sent to the burn wallet (0x00dEaD) they will instead be removed from the senders balance and removed from the total supply.

When this is set to false, any tokens sent to the burn wallet will not be removed from total supply and will be added to the burn wallet balance. This is the default action on most contracts.

A truly deflationary burn can be confusing to some token tools and listing platforms, so only set this to true if you understand the implications.

A deflationary burn will not instantly increase the value of

*other tokens,
but it will help with token stability over time.*

**/*

```
function Contract_Options_01__Deflationary_Burn(bool  
true_or_false) external onlyOwner {  
    deflationaryBurn = true_or_false;  
}
```

*/**

*-----
DECIDE IF WALLET TO WALLET TRANSFERS WILL BE FREE FROM FEES
-----*

Default = true

*Having no fee on wallet-to-wallet transfers means that people
can move tokens between wallets,
or send them to friends etc without incurring a fee.*

*This feature may be required if you plan to use your token in
place of fiat as a form of payment.*

*However, in order for it to work, we must inform the contract
of all liquidity pairs. So*

*if you (or anybody else) ever adds a new liquidity pair, you
need to enter the address of the pair
into the "Maintenance_02__Add_Liquidity_Pair" function.*

*If you plan to renounce your contract, you will lose access to
all functions. Which presents a
possible exploit where people can create a liquidity pair for
your token and use it to buy and sell
without a fee.*

*For this reason, you can not renounce the contract and have
no-fee on wallet-to-wallet transfers.*

Decide which is better for your project. No fees when moving tokens between wallets, or renouncing ownership. Having both is not an option!

**/*

```
function Contract_Options_02__No_Fee_Wallet_Transfers(bool
true_or_false) public onlyOwner {
    noFeeW2W = true_or_false;
}
```

*/**

SET CONTRACT BUY AND SELL FEES

To protect investors, buy and sell fees have a hard-coded limit of 20%

If the contract development fee was set to 1% of transactions, this is included in the limit.

How Fees Work

Burn, Token, and Reflection fees are processed immediately during the transaction.

BNB and Liquidity fees are collected in tokens then added to the contract.

These fees accumulate (as tokens) on the contract until they are processed.

When fees are processed, the contract sells the accumulated tokens for BNB

(This shows as a sell on the chart).

This process can only happen when a holder sells tokens.

So when fees are processed, you will see 2 sells on the chart in the same second, the holders sell, and the contract sell.

This process is triggered automatically on the next sell after 10 transactions.

**/*

// Set Buy Fees

function Contract_SetUp_01__Fees_on_Buy(

uint256 BNB_on_BUY,
uint256 Liquidity_on_BUY,
uint256 Reflection_on_BUY,
uint256 Burn_on_BUY,
uint256 Tokens_on_BUY

) external onlyOwner {

_Fee__Buy_Contract = ContractFee;

*// Buyer protection: max fee can not be set over 20%
(including the 1% contract fee if applicable)*

require (BNB_on_BUY +
Liquidity_on_BUY +
Reflection_on_BUY +
Burn_on_BUY +
Tokens_on_BUY +
_Fee__Buy_Contract <= 20, "E02");

// Update fees

_Fee__Buy_BNB = BNB_on_BUY;
_Fee__Buy_Liquidity = Liquidity_on_BUY;
_Fee__Buy_Reflection = Reflection_on_BUY;
_Fee__Buy_Burn = Burn_on_BUY;
_Fee__Buy_Tokens = Tokens_on_BUY;

```

        // Fees that will need to be processed during swap and
        Liquify
        _SwapFeeTotal_Buy      = _Fee__Buy_BNB + _Fee__Buy_Liquidity
+ _Fee__Buy_Contract;

        emit updated_Buy_fees(_Fee__Buy_BNB, _Fee__Buy_Liquidity,
        _Fee__Buy_Reflection, _Fee__Buy_Burn, _Fee__Buy_Tokens,
        _Fee__Buy_Contract);
    }

    // Set Sell Fees
    function Contract_SetUp_02__Fees_on_Sell(

        uint256 BNB_on_SELL,
        uint256 Liquidity_on_SELL,
        uint256 Reflection_on_SELL,
        uint256 Burn_on_SELL,
        uint256 Tokens_on_SELL

    ) external onlyOwner {

        _Fee__Sell_Contract = ContractFee;

        // Buyer protection: max fee can not be set over 20%
        (including the 1% contract fee if applicable)
        require (BNB_on_SELL      +
            Liquidity_on_SELL  +
            Reflection_on_SELL +
            Burn_on_SELL      +
            Tokens_on_SELL    +
            _Fee__Sell_Contract <= 20, "E03");

        // Update fees
        _Fee__Sell_BNB      = BNB_on_SELL;
        _Fee__Sell_Liquidity = Liquidity_on_SELL;
        _Fee__Sell_Reflection = Reflection_on_SELL;
        _Fee__Sell_Burn      = Burn_on_SELL;
        _Fee__Sell_Tokens    = Tokens_on_SELL;

        // Fees that will need to be processed during swap and
        Liquify
        _SwapFeeTotal_Sell    = _Fee__Sell_BNB +

```



```

_Fee__Sell_Liquidity + _Fee__Sell_Contract;

        emit updated_Sell_fees(_Fee__Sell_BNB,
_Fee__Sell_Liquidity, _Fee__Sell_Reflection, _Fee__Sell_Burn,
_Fee__Sell_Tokens, _Fee__Sell_Contract);
    }

    /*

    -----
    SET MAX TRANSACTION AND MAX HOLDING LIMITS
    -----

    To protect buyers, these values must be set to a minimum of
    0.1% of the total supply

    Wallet limits are set as a number of tokens, not as a percent
    of supply!

    If you want to limit people to 2% of supply and your supply is
    1,000,000 tokens then you
    will need to enter 20000 (as this is 2% of 1,000,000)

    */

    // Wallet Holding and Transaction Limits (Enter token amount,
    excluding decimals)
    function Contract_SetUp_03__Wallet_Limits(

        uint256 Max_Tokens_Per_Transaction,
        uint256 Max_Total_Tokens_Per_Wallet

    ) external onlyOwner {

        // Buyer protection - Limits must be set to greater than
        0.1% of total supply
        require(Max_Tokens_Per_Transaction >= _tTotal / 1000 /
10**_decimals, "E04");
        require(Max_Total_Tokens_Per_Wallet >= _tTotal / 1000 /
10**_decimals, "E05");
    }

```

```

    max-Tran = Max_Tokens_Per_Transaction * 10**_decimals;
    max-Hold = Max_Total_Tokens_Per_Wallet * 10**_decimals;

    emit updated_Wallet_Limits(max-Tran, max-Hold);

}

/*

-----
UPDATE PROJECT WALLETS
-----

The contract can process fees in the native token or BNB.
Processed fees are sent to external wallets (Token_Fee_Wallet
and BNB_Fee_Wallet).

Cake LP Tokens that are created when the contract makes Auto
Liquidity are sent to the Liquidity_Collection_Wallet
Periodically, these tokens will need to be Locked (or burned).

During deployment, all external wallets are set to the owner
wallet by default, but can be updated here.

INVESTORS - Please check the project website for details of
how fees are distributed.

*/

function Contract_SetUp_04__Set_Wallets(

    address Token_Fee_Wallet,
    address payable BNB_Fee_Wallet,
    address Liquidity_Collection_Wallet

) external onlyOwner {

    // Update Token Fee Wallet
    require(Token_Fee_Wallet != address(0), "E06");

```

```

    Wallet_Tokens = Token_Fee_Wallet;

    // Make limit exempt
    _isLimitExempt[Token_Fee_Wallet] = true;

    // Update BNB Fee Wallet
    require(BNB_Fee_Wallet != address(0), "E07");
    Wallet_BNB = payable(BNB_Fee_Wallet);

    // To send the auto liquidity tokens directly to burn
    update to 0x0000000000000000000000000000000000000000dEaD
    Wallet_Liquidity = Liquidity_Collection_Wallet;

}

/*

-----
ADD PROJECT LINKS
-----

The information that you add here will appear on BSCScan,
helping potential investors to find out more about your project.
Be sure to enter the complete URL as many websites will
automatically detect this, and add links to your token listing.

If you are updating one link, you will also need to re-enter
the other two links.

*/

function Contract_SetUp_05__Update_Socials(

    string memory Website_URL,
    string memory Telegram_URL,
    string memory Liquidity_Locker_URL

) external onlyOwner{

    _Website          = Website_URL;
    _Telegram          = Telegram_URL;
    _LP_Locker_URL    = Liquidity_Locker_URL;

```

```
}
```

```
/*
```

```
-----  
SET UP PRE-SALE CONTRACT ADDRESS  
-----
```

If you are doing a pre-sale, the pre-sale company will give you an address and tell you that it needs to be white-listed.

Enter it here and it will be granted the required privileges.

Do not continue with contract setup until the pre-sale has been finalized.

```
*/
```

```
function Contract_SetUp_06__PreSale_Wallet (address  
PreSale_Wallet_Address) external onlyOwner {
```

```
    _isLimitExempt[PreSale_Wallet_Address]      = true;  
    _isExcludedFromFee[PreSale_Wallet_Address]   = true;  
    _isWhiteListed[PreSale_Wallet_Address]      = true;
```

```
}
```

```
/*
```

```
-----  
BLACKLIST BOTS - PRE LAUNCH ONLY!  
-----
```

*You have the ability to blacklist wallets prior to launch.
This should only be used for known bot users.*

Check <https://poocoin.app/sniper-watcher> to see currently active bot users

*To blacklist, enter a wallet address and set to true.
To remove blacklist, enter a wallet address and set to false.*

To protect your investors (and improve your audit score) you can only blacklist wallets before public launch. However, you will still be able to 'un-blacklist' previously blacklisted wallets after launch.

```
*/  
  
function Launch_Settings_01__Blacklist_Bots(  
  
    address Wallet,  
    bool true_or_false  
  
    ) external onlyOwner {  
  
    // Buyer Protection - Blacklisting can only be done before  
Launch  
    if (true_or_false){require(LaunchTime == 0, "E08");}  
    _isBlacklisted[Wallet] = true_or_false;  
}  
  
/*  
  
-----  
SET LAUNCH LIMIT RESTRICTIONS  
-----
```

During the launch phase, additional restrictions can help to spread the tokens more evenly over the initial buyers.

This helps to prevent whales accumulating a max wallet for almost nothing and prevent dumps.

Settings:

Launch_Buy_Delay_Seconds = Number of seconds a buyer will have to wait before buying again

Launch_Transaction_Limit = Amount of TOKENS that can be purchased in one transaction

Launch_Phase_Length_Minutes = Time (in minutes) that launch phase restrictions will last

Important:

Remember that the transaction limit is in TOKENS not a percent of total supply!

Recommendations:

I'd suggest having a delay timer of 10 to 20 seconds, a transaction limit of 50% of your standard transaction limit, and a launch phase length of about 5 minutes

**/*

```
function Launch_Settings_02__Set_Launch_Limits(

    uint256 Launch_Buy_Delay_Seconds,
    uint256 Launch_Transaction_Limit_TOKENS,
    uint256 Launch_Phase_Length_Minutes

) external onlyOwner {

    max_Tran_Launch = Launch_Transaction_Limit_TOKENS * 10 **
_decimals;
    Launch_Buy_Delay = Launch_Buy_Delay_Seconds;
    Launch_Length = Launch_Phase_Length_Minutes * 60;
```

```
}
```

```
/*
```

```
-----  
ADD LIQUIDITY  
-----
```

If you have done a pre-sale, the pre-sale company will most likely add the liquidity for you automatically. If you are not doing a pre-sale, but you plan to do a private sale, you must add the liquidity now, but do not open trade until the private sale is complete.

To add your liquidity go to <https://pancakeswap.finance/add/BNB> and enter your contract address into the 'Select' field.

```
-----  
COMPLETE AIRDROPS  
-----
```

If your project requires that you airdrop people tokens, you should do this after adding liquidity. This will prevent any whitelisted token holders from adding liquidity before you and thus setting the price of your token.

```
*/
```



```

/*
-----
OPEN TRADE
-----

*/

// Open trade: Buyer Protection - one way switch - trade can
not be paused once opened
function Launch_Settings_03__OpenTrade() external onlyOwner {

    // Can only use once!
    require(!TradeOpen, "E09");
    TradeOpen = true;
    swapAndLiquifyEnabled = true;
    LaunchPhase = true;
    LaunchTime = block.timestamp;

    emit updated_trade_Open(TradeOpen);
    emit
updated_SwapAndLiquify_Enabled(swapAndLiquifyEnabled);

    // Set the contract fee if required
    _Fee__Buy_Contract = ContractFee;
    _Fee__Sell_Contract = ContractFee;
    _SwapFeeTotal_Buy = _Fee__Buy_Liquidity + _Fee__Buy_BNB
+ _Fee__Buy_Contract;
    _SwapFeeTotal_Sell = _Fee__Sell_Liquidity +
_Fee__Sell_BNB + _Fee__Sell_Contract;

    emit set_Contract_Fee(_Fee__Buy_Contract,
_Fee__Sell_Contract);
}

```

```

/*

-----
CONTRACT MAINTENANCE FUNCTIONS
-----

*/


/*

-----
REMOVE CONTRACT FEE
-----

Remove 1% Contract Fee for 2 BNB

If you opted for the 1% ongoing fee in your contract you can
remove this at a cost of 2 BNB at any time.
To do this, enter the number 2 into the field.

WARNING - If you renounce the contract, you will lose access
to this function!

*/

function Maintenance_01__Remove_Contract_Fee() external
onlyOwner payable {

    require(msg.value == 2*10**18, "E10");

    // Check Affiliate is genuine - (Holds the TokensByGEN
Affiliate Token)
    if(AFT.balanceOf(Wallet_TBG_AFF) > 0){

        uint256 AFF_2BNB = 0;
        uint256 TBG_2BNB = 0;

        // Calculate the affiliate percentage
based on GEN holding

```

```

        if(GEN.balanceOf(Wallet_TBG_AFF) >=
Tier_3){

            AFF_2BNB = 20;
            TBG_2BNB = 80;

        } else if
(GEN.balanceOf(Wallet_TBG_AFF) >= Tier_2){

            AFF_2BNB = 15;
            TBG_2BNB = 85;

        } else {

            AFF_2BNB = 10;
            TBG_2BNB = 90;

        }

        // Send BNB to affiliate and
TokensByGEN Contract Fee
        if (AFF_2BNB > 0){

            send_BNB(Wallet_TBG_AFF, msg.value
* AFF_2BNB / 100);

            send_BNB(feeCollector, msg.value *
TBG_2BNB / 100);

        }

    } else {

        // Affiliate is not valid, send BNB to TokensByGEN
contract Fee only
        send_BNB(feeCollector, msg.value);

    }

    // Remove Contract Fee
    ContractFee          = 0;
    _Fee__Buy_Contract   = 0;
    _Fee__Sell_Contract   = 0;

```

```

        // Emit Contract Fee update
        emit set_Contract_Fee(_Fee__Buy_Contract,
        _Fee__Sell_Contract);

        // Update Swap Fees
        _SwapFeeTotal_Buy    = _Fee__Buy_Liquidity + _Fee__Buy_BNB;
        _SwapFeeTotal_Sell   = _Fee__Sell_Liquidity +
        _Fee__Sell_BNB;
    }

```

/*

 ADDING A NEW LIQUIDITY PAIR

The only way that your contract knows to apply a fee is when an address is set as true via this function.

This has already been done for your BNB pair, but if you add a new pair you need to enter the address of that pair into this function and set it to true.

When you create a new liquidity pair on pancake swap they mint a new token (called Cake LP) with a unique address that represents your token and the other token you used to create the pool.

Remember that anybody can create a new liquidity pair for any token. So if you renounce ownership, you will lose the ability to update the contract with the new pair address.

If you have no-fee for wallet-to-wallet transfers (the default) then there is a potential exploit where the new liquidity pair could be used to purchase tokens without paying a fee.

Therefore, if you plan to renounce, you must first deactivate the no fee option for wallet-to-wallet transfers.

You can do this using the "Contract_Options_02__No_Fee_Wallet_Transfers" function.

Obviously, this is something you need to be very transparent about. If you tell people your token has no fee for wallet transfers and later change this, you could be responsible for people losing money.

It is best to decide from the very beginning if you plan to renounce in future. If you do, then immediately deactivate the fee-free transfer option and do not promote it as a feature of your token.

```
*/

// Setting an address as a liquidity pair
function Maintenance_02__Add_Liquidity_Pair(

    address Wallet_Address,
    bool true_or_false)

    external onlyOwner {
        _isPair[Wallet_Address] = true_or_false;
        _isLimitExempt[Wallet_Address] = true_or_false;
    }

}

/*

-----
CONTRACT OWNERSHIP FUNCTIONS
-----

*/

// Transfer the contract to to a new owner
function Maintenance_03__Transfer_Ownership(address payable
```

```

newOwner) public onlyOwner {
    require(newOwner != address(0), "E11");

    // Remove old owner status
    _isLimitExempt[owner()] = false;
    _isExcludedFromFee[owner()] = false;
    _isWhiteListed[owner()] = false;

    // Emit ownership transfer
    emit OwnershipTransferred(_owner, newOwner);

    // Transfer owner
    _owner = newOwner;
}

```

```

/*

```

Due to a potential exploit, it is not possible to renounce the contract while no-fee wallet-to-wallet transfers are set to true. To deactivate this option, use the "Contract_Options_02__No_Fee_Wallet_Transfers" function and set it as 'false' before renouncing.

```

*/

```

```

// Renounce ownership of the contract
function Maintenance_04__Renounce_Ownership() public virtual
onlyOwner {
    // Renouncing is not compatible with no-fee
    // wallet-to-wallet transfers
    // (also prevents 'accidental' renounce... People like
    // clicking buttons!)
    require(!noFeeW2W, "Can not renounce and have no-fee
wallet transfers!");
    // Remove old owner status
    _isLimitExempt[owner()] = false;
    _isExcludedFromFee[owner()] = false;
    _isWhiteListed[owner()] = false;
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}

```

```

}

/*

-----
FEE PROCESSING
-----

*/

// Default is True. Contract will process fees into Marketing
and Liquidity etc. automatically
function Processing_01__Auto_Process(bool true_or_false)
external onlyOwner {
    swapAndLiquifyEnabled = true_or_false;
    emit updated_SwapAndLiquify_Enabled(true_or_false);
}

// Manually process fees
function Processing_02__Process_Now (uint256
Percent_of_Tokens_to_Process) external onlyOwner {
    require(!inSwapAndLiquify, "E12");
    if (Percent_of_Tokens_to_Process >
100){Percent_of_Tokens_to_Process == 100;}
    uint256 tokensOnContract = balanceOf(address(this));
    uint256 sendTokens = tokensOnContract *
Percent_of_Tokens_to_Process / 100;
    swapAndLiquify(sendTokens);
}

// Update count for swap trigger - Number of transactions to
wait before processing accumulated fees (default is 10)
function Processing_03__Update_Swap_Trigger_Count(uint256

```



```

Transaction_Count) external onlyOwner {
    // Counter is reset to 1 (not 0) to save gas, so add one
    to swapTrigger
    swapTrigger = Transaction_Count + 1;
}

// Remove random tokens from the contract
function Processing_04__Remove_Random_Tokens(

    address random_Token_Address,
    uint256 number_of_Tokens

) external onlyOwner {
    // Can not purge the native token!
    require (random_Token_Address != address(this),
"E13");
    IERC20(random_Token_Address).transfer(msg.sender,
number_of_Tokens);
}

/*

-----
REFLECTION REWARDS
-----

    The following functions are used to exclude or include a
    wallet in the reflection rewards.
    By default, all wallets are included.

    Wallets that are excluded:

        The Burn address
        The Liquidity Pair
        The Contract Address

-----
*** WARNING - DoS 'OUT OF GAS' Risk! ***
-----

```

A reflections contract needs to loop through all excluded wallets to correctly process several functions.

This loop can break the contract if it runs out of gas before completion.

To prevent this, keep the number of wallets that are excluded from rewards to an absolute minimum.

In addition to the default excluded wallets, you may need to exclude the address of any locked tokens.

```
*/

// Wallet will not get reflections
function Rewards_Exclude_Wallet(address account) public
onlyOwner() {
    require(!_isExcluded[account], "Account is already
excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] =
tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

// Wallet will get reflections - DEFAULT
function Rewards_Include_Wallet(address account) external
onlyOwner() {
    require(_isExcluded[account], "Account is already
included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

```

}

/*

-----
WALLET SETTINGS
-----

*/

// Grants access when trade is closed - Default false (true
for contract owner)
function Wallet_Settings_01__PreLaunch_Access(

    address Wallet_Address,
    bool true_or_false

) external onlyOwner {
    _isWhiteListed[Wallet_Address] = true_or_false;
}

// Excludes wallet from transaction and holding limits -
Default false
function Wallet_Settings_02__Exempt_From_Limits(

    address Wallet_Address,
    bool true_or_false

) external onlyOwner {
    _isLimitExempt[Wallet_Address] = true_or_false;
}

// Excludes wallet from fees - Default false
function Wallet_Settings_03__Exclude_From_Fees(

```

```

    address Wallet_Address,
    bool true_or_false

    ) external onlyOwner {
        _isExcludedFromFee[Wallet_Address] = true_or_false;
    }

/*

-----
BEP20 STANDARD AND COMPLIANCE
-----

*/

function owner() public view returns (address) {
    return _owner;
}

function name() public view returns (string memory) {
    return _name;
}

function symbol() public view returns (string memory) {
    return _symbol;
}

function decimals() public view returns (uint256) {
    return _decimals;
}

function totalSupply() public view override returns (uint256)

```

```

{
    return _tTotal;
}

function balanceOf(address account) public view override
returns (uint256) {
    if (_isExcluded[account]) return _tOwned[account];
    return tokenFromReflection(_rOwned[account]);
}

function allowance(address owner, address spender) public view
override returns (uint256) {
    return _allowances[owner][spender];
}

function increaseAllowance(address spender, uint256
addedValue) public virtual returns (bool) {
    _approve(_msgSender(), spender,
_allowances[_msgSender()][spender].add(addedValue));
    return true;
}

function decreaseAllowance(address spender, uint256
subtractedValue) public virtual returns (bool) {
    _approve(_msgSender(), spender,
_allowances[_msgSender()][spender].sub(subtractedValue, "Decreased
allowance below zero"));
    return true;
}

function approve(address spender, uint256 amount) public
override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function _approve(address owner, address spender, uint256
amount) private {
    require(owner != address(0), "BEP20: approve from the zero
address");
    require(spender != address(0), "BEP20: approve to the zero
address");
}

```

```

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    function tokenFromReflection(uint256 _rAmount) internal view
    returns(uint256) {
        require(_rAmount <= _rTotal, "E14");
        uint256 currentRate = _getRate();
        return _rAmount / currentRate;
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply / tSupply;
    }

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

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

    function transferFrom(address sender, address recipient,
    uint256 amount) public override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(),

```

```

_allowances[sender][_msgSender()].sub(amount, "Allowance
exceeded"));
    return true;
}

// Transfer BNB via call to reduce possibility of future 'out
of gas' errors
function send_BNB(address _to, uint256 _amount) internal
returns (bool SendSuccess) {

    (SendSuccess,) = payable(_to).call{value: _amount}("");
}

/*

-----
TOKEN TRANSFER HANDLING
-----

*/

// Main transfer checks and settings
function _transfer(
    address from,
    address to,
    uint256 amount
) private {

    // Allows owner to add liquidity safely, eliminating the
risk of someone maliciously setting the price
    if (!TradeOpen){
        require(_isWhiteListed[from] || _isWhiteListed[to],

```

```

"E15");
    }

    // Launch Phase
    if (LaunchPhase && to != address(this) && _isPair[from] &&
to != owner())
    {

        // Restrict max transaction during launch phase
        require(amount <= max-Tran-Launch, "E16");

        // Stop repeat buys with timer
        require (block.timestamp >= _Last_Buy[to] +
Launch_Buy_Delay, "E17");

        // Stop snipers
        require(!_isSnipe[to], "E18");

        // Detect and restrict snipers
        if (block.timestamp <= LaunchTime + 5) {
            require(amount <= _tTotal / 10000, "E19");
            _isSnipe[to] = true;
        }

        // Record the transaction time for the buying wallet
        _Last_Buy[to] = block.timestamp;

        // End Launch Phase after Launch_Length (minutes)
        if (block.timestamp > LaunchTime +
Launch_Length){LaunchPhase = false;}

    }

    // No blacklisted wallets permitted!
    require(!_isBlacklisted[to] &&
!_isBlacklisted[from], "E20");

    // Wallet Limit

```



```

    if (!_isLimitExempt[to] && from != owner())
    {
        uint256 heldTokens = balanceOf(to);
        require((heldTokens + amount) <= max_Hold, "E21");
    }

    // Transaction Limit - To send over the transaction limit
the sender AND the recipient must be limit exempt
    if (!_isLimitExempt[to] || !_isLimitExempt[from])
    {
        require(amount <= max_Trans, "E22");
    }

    // Compliance and safety checks
    require(from != address(0), "E23");
    require(to != address(0), "E24");
    require(amount > 0, "E25");

    // Check if fee processing is possible
    if( _isPair[to] &&
        !inSwapAndLiquify &&
        swapAndLiquifyEnabled
    )
    {

        // Check that enough transactions have passed since
last swap
        if(swapCounter >= swapTrigger){

            // Check number of tokens on contract
            uint256 contractTokens = balanceOf(address(this));

            // Only trigger fee processing if there are tokens to
swap!
            if (contractTokens > 0){

                // Limit number of tokens that can be swapped
                if (contractTokens <= swap_Max){

```

```

        swapAndLiquify (contractTokens);
    } else {
        swapAndLiquify (swap_Max);
    }
}
}
}

    // Default: Only charge a fee on buys and sells, no fee
for wallet transfers
    takeFee = true;
    if(!_isExcludedFromFee[from] || _isExcludedFromFee[to] ||
(noFeeW2W && !_isPair[to] && !_isPair[from])){
        takeFee = false;
    }

    _tokenTransfer(from, to, amount, takeFee);

}

/*

-----
PROCESS FEES
-----

*/

function swapAndLiquify(uint256 Tokens) private {

    /*

    Fees are processed as an average of each buy/sell fee
total

    */

    // Lock swapAndLiquify function
inSwapAndLiquify      = true;

```

```

        uint256 _FeesTotal      = (_SwapFeeTotal_Buy +
_SwapFeeTotal_Sell);
        uint256 LP_Tokens      = Tokens * (_Fee__Buy_Liquidity +
_Fee__Sell_Liquidity) / _FeesTotal / 2;
        uint256 Swap_Tokens    = Tokens - LP_Tokens;

        // Swap tokens for BNB
        uint256 contract_BNB    = address(this).balance;
        swapTokensForBNB(Swap_Tokens);
        uint256 returned_BNB    = address(this).balance -
contract_BNB;

        // Double fees instead of halving LP fee to prevent
rounding errors if fee is an odd number
        uint256 fee_Split = _FeesTotal * 2 - (_Fee__Buy_Liquidity
+ _Fee__Sell_Liquidity);

        // Calculate the BNB values for each fee (excluding BNB
wallet)
        uint256 BNB_Liquidity   = returned_BNB *
(_Fee__Buy_Liquidity      + _Fee__Sell_Liquidity)      /
fee_Split;
        uint256 BNB_Contract    = returned_BNB *
(_Fee__Buy_Contract      + _Fee__Sell_Contract)      * 2 /
fee_Split;

        // Add Liquidity
        if (LP_Tokens != 0){
            addLiquidity(LP_Tokens, BNB_Liquidity);
            emit SwapAndLiquify(LP_Tokens, BNB_Liquidity,
LP_Tokens);
        }

        // Take developer fee
        if(BNB_Contract > 0){

            // Check Affiliate is genuine - (Holds the TokensByGEN
Affiliate Token)
            if(AFT.balanceOf(Wallet_TBG_AFF) > 0){

                uint256 BNB_TBG = 0;

```

```

uint256 BNB_DEV = 0;

if(GEN.balanceOf(Wallet_TBG_AFF) >= Tier_3){

    BNB_TBG = BNB_Contract * 20 / 100;
    BNB_DEV = BNB_Contract * 70 / 100;

} else if (GEN.balanceOf(Wallet_TBG_AFF) >=
Tier_2){

    BNB_TBG = BNB_Contract * 15 / 100;
    BNB_DEV = BNB_Contract * 75 / 100;

} else {

    BNB_TBG = BNB_Contract * 10 / 100;
    BNB_DEV = BNB_Contract * 80 / 100;

}

if (BNB_TBG != 0){

    // Send affiliate commission and
TokensByGEN fee

    send_BNB(Wallet_TBG_AFF, BNB_TBG);
    send_BNB(feeCollector, BNB_DEV);

}

} else {

    // No affiliate (or not genuine) send total fee to
TokensByGEN

    send_BNB(feeCollector, BNB_Contract);

}

}

// Send remaining BNB to BNB wallet (includes 10% fee
discount if applicable)

```

```

        contract_BNB = address(this).balance;

        if(contract_BNB > 0){

            send_BNB(Wallet_BNB, contract_BNB);
        }

        // Reset transaction counter (reset to 1 not 0 to save
gas)
        swapCounter = 1;

        // Unlock swapAndLiquify function
        inSwapAndLiquify = false;
    }

    // Swap tokens for BNB
    function swapTokensForBNB(uint256 tokenAmount) private {

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        _approve(address(this), address(uniswapV2Router),
tokenAmount);

        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
        (
            tokenAmount,
            0,
            path,
            address(this),
            block.timestamp
        );
    }

    // Add liquidity and send Cake LP tokens to liquidity
collection wallet
    function addLiquidity(uint256 tokenAmount, uint256 BNBAmount)

```

```

private {

    _approve(address(this), address(uniswapV2Router),
tokenAmount);
    uniswapV2Router.addLiquidityETH{value: BNBAmount}(
        address(this),
        tokenAmount,
        0,
        0,
        Wallet_Liquidity,
        block.timestamp
    );
}

```

```

/*

```

```

-----
TRANSFER TOKENS AND CALCULATE FEES
-----

```

```

*/

```

```

uint256 private rAmount;

```

```

uint256 private tBurn;

```

```

uint256 private tTokens;

```

```

uint256 private tReflect;

```

```

uint256 private tSwapFeeTotal;

```

```

uint256 private rBurn;

```

```

uint256 private rReflect;

```

```

uint256 private rTokens;

```

```

uint256 private rSwapFeeTotal;

```

```

uint256 private tTransferAmount;

```

```

uint256 private rTransferAmount;

// Transfer Tokens and Calculate Fees
function _tokenTransfer(address sender, address recipient,
uint256 tAmount, bool Fee) private {

    if (Fee){

        if(_isPair[recipient]){

            // Sell fees
            tBurn          = tAmount * _Fee__Sell_Burn
/ 100;
            tTokens        = tAmount * _Fee__Sell_Tokens
/ 100;
            tReflect        = tAmount * _Fee__Sell_Reflection
/ 100;
            tSwapFeeTotal   = tAmount * _SwapFeeTotal_Sell
/ 100;

        } else {

            // Buy fees
            tBurn          = tAmount * _Fee__Buy_Burn
/ 100;
            tTokens        = tAmount * _Fee__Buy_Tokens
/ 100;
            tReflect        = tAmount * _Fee__Buy_Reflection
/ 100;
            tSwapFeeTotal   = tAmount * _SwapFeeTotal_Buy
/ 100;

        }

    } else {

        // No fee - wallet to wallet transfer or exempt
wallet
        tBurn              = 0;

```

```

        tTokens          = 0;
        tReflect         = 0;
        tSwapFeeTotal    = 0;

    }

    // Calculate reflected fees for RFI
    uint256 RFI          = _getRate();

    rAmount              = tAmount          * RFI;
    rBurn                = tBurn            * RFI;
    rTokens              = tTokens          * RFI;
    rReflect              = tReflect         * RFI;
    rSwapFeeTotal        = tSwapFeeTotal * RFI;

    tTransferAmount = tAmount - (tBurn + tTokens + tReflect +
tSwapFeeTotal);
    rTransferAmount = rAmount - (rBurn + rTokens + rReflect +
rSwapFeeTotal);

    // Swap tokens based on RFI status of sender and recipient
    if (_isExcluded[sender] && !_isExcluded[recipient]) {

        _tOwned[sender] -= tAmount;
        _rOwned[sender] -= rAmount;

        if (deflationaryBurn && recipient == Wallet_Burn)
{

            // Remove tokens from Total Supply
            _tTotal -= tTransferAmount;
            _rTotal -= rTransferAmount;

        } else {

            _rOwned[recipient] += rTransferAmount;

        }

        emit Transfer(sender, recipient, tTransferAmount);

```



```
    } else if (!_isExcluded[sender] && _isExcluded[recipient])
    {

        _rOwned[sender] -= rAmount;

        if (deflationaryBurn && recipient == Wallet_Burn)
        {

            // Remove tokens from Total Supply
            _tTotal -= tTransferAmount;
            _rTotal -= rTransferAmount;

        } else {

            _tOwned[recipient] += tTransferAmount;
            _rOwned[recipient] += rTransferAmount;

        }

        emit Transfer(sender, recipient, tTransferAmount);

    } else if (!_isExcluded[sender] &&
!_isExcluded[recipient]) {

        _rOwned[sender] -= rAmount;

        if (deflationaryBurn && recipient == Wallet_Burn)
        {

            // Remove tokens from Total Supply
            _tTotal -= tTransferAmount;
            _rTotal -= rTransferAmount;

        } else {

            _rOwned[recipient] += rTransferAmount;

        }

        emit Transfer(sender, recipient, tTransferAmount);

    } else if (_isExcluded[sender] && _isExcluded[recipient])
```

```

{

    _tOwned[sender] -= tAmount;
    _rOwned[sender] -= rAmount;

    if (deflationaryBurn && recipient == Wallet_Burn)
{

    // Remove tokens from Total Supply
    _tTotal -= tTransferAmount;
    _rTotal -= rTransferAmount;

    } else {

        _tOwned[recipient] += tTransferAmount;
        _rOwned[recipient] += rTransferAmount;

    }

    emit Transfer(sender, recipient, tTransferAmount);
} else {
    _rOwned[sender] -= rAmount;

    if (deflationaryBurn && recipient == Wallet_Burn)
{

    // Remove tokens from Total Supply
    _tTotal -= tTransferAmount;
    _rTotal -= rTransferAmount;

    } else {

        _rOwned[recipient] += rTransferAmount;

    }

    emit Transfer(sender, recipient, tTransferAmount);
}
// Take reflections
if(tReflect > 0){

    _rTotal -= rReflect;

```

```

        _tFeeTotal += tReflect;
    }

    // Take tokens
    if(tTokens > 0){

        _rOwned[Wallet_Tokens] += rTokens;
        if(!_isExcluded[Wallet_Tokens])
            _tOwned[Wallet_Tokens] += tTokens;
    }

    // Take fees that require processing during swap and
    // Liquify
    if(tSwapFeeTotal > 0){

        _rOwned[address(this)] += rSwapFeeTotal;
        if(!_isExcluded[address(this)])
            _tOwned[address(this)] += tSwapFeeTotal;

        // Increase the transaction counter
        swapCounter++;
    }

    // Handle tokens for burn
    if(tBurn != 0){
        if (deflationaryBurn){
            // Remove tokens from total supply
            _tTotal = _tTotal - tBurn;
            _rTotal = _rTotal - rBurn;
        } else {

            // Send Tokens to Burn Wallet
            _rOwned[Wallet_Burn] += tBurn;
            if(!_isExcluded[Wallet_Burn])
                _tOwned[Wallet_Burn] += rBurn;
        }
    }
}

```

4. Tax Fee contract

```
// Set Buy Fees
function Contract_SetUp_01__Fees_on_Buy(

    uint256 BNB_on_BUY,
    uint256 Liquidity_on_BUY,
    uint256 Reflection_on_BUY,
    uint256 Burn_on_BUY,
    uint256 Tokens_on_BUY

) external onlyOwner {

    _Fee__Buy_Contract = ContractFee;

    // Buyer protection: max fee can not be set over 20%
    (including the 1% contract fee if applicable)
    require (BNB_on_BUY      +
             Liquidity_on_BUY +
             Reflection_on_BUY +
             Burn_on_BUY     +
             Tokens_on_BUY    +
             _Fee__Buy_Contract <= 20, "E02");

    // Update fees
    _Fee__Buy_BNB      = BNB_on_BUY;
    _Fee__Buy_Liquidity = Liquidity_on_BUY;
    _Fee__Buy_Reflection = Reflection_on_BUY;
    _Fee__Buy_Burn      = Burn_on_BUY;
    _Fee__Buy_Tokens    = Tokens_on_BUY;

    // Fees that will need to be processed during swap and
    Liquify
    _SwapFeeTotal_Buy    = _Fee__Buy_BNB + _Fee__Buy_Liquidity
+ _Fee__Buy_Contract;

    emit updated_Buy_fees(_Fee__Buy_BNB, _Fee__Buy_Liquidity,
    _Fee__Buy_Reflection, _Fee__Buy_Burn, _Fee__Buy_Tokens,
    _Fee__Buy_Contract);
}

// Set Sell Fees
function Contract_SetUp_02__Fees_on_Sell(
```

```

uint256 BNB_on_SELL,
uint256 Liquidity_on_SELL,
uint256 Reflection_on_SELL,
uint256 Burn_on_SELL,
uint256 Tokens_on_SELL

) external onlyOwner {

    _Fee__Sell_Contract = ContractFee;

    // Buyer protection: max fee can not be set over 20%
(including the 1% contract fee if applicable)
    require (BNB_on_SELL      +
             Liquidity_on_SELL +
             Reflection_on_SELL +
             Burn_on_SELL      +
             Tokens_on_SELL    +
             _Fee__Sell_Contract <= 20, "E03");

    // Update fees
    _Fee__Sell_BNB      = BNB_on_SELL;
    _Fee__Sell_Liquidity = Liquidity_on_SELL;
    _Fee__Sell_Reflection = Reflection_on_SELL;
    _Fee__Sell_Burn      = Burn_on_SELL;
    _Fee__Sell_Tokens    = Tokens_on_SELL;

    // Fees that will need to be processed during swap and
liquify
    _SwapFeeTotal_Sell  = _Fee__Sell_BNB +
    _Fee__Sell_Liquidity + _Fee__Sell_Contract;

    emit updated_Sell_fees(_Fee__Sell_BNB,
    _Fee__Sell_Liquidity, _Fee__Sell_Reflection, _Fee__Sell_Burn,
    _Fee__Sell_Tokens, _Fee__Sell_Contract);
}

```

The owner can't set fees over 20%

5. Max Tx Amount

```
function Contract_SetUp_03__Wallet_Limits(

    uint256 Max_Tokens_Per_Transaction,
    uint256 Max_Total_Tokens_Per_Wallet

) external onlyOwner {

    // Buyer protection - Limits must be set to greater than
0.1% of total supply
    require(Max_Tokens_Per_Transaction >= _tTotal / 1000 /
10**_decimals, "E04");
    require(Max_Total_Tokens_Per_Wallet >= _tTotal / 1000 /
10**_decimals, "E05");

    max_Tran = Max_Tokens_Per_Transaction * 10**_decimals;
    max_Hold = Max_Total_Tokens_Per_Wallet * 10**_decimals;

    emit updated_Wallet_Limits(max_Tran, max_Hold);

}
```

Owner can't set max tx amount below 0.1% from total supply
Owner can't set max amount hold below 0.1% from total supply

6. Blacklist address - Medium Risk

```
function Launch_Settings_01__Blacklist_Bots(

    address Wallet,
    bool true_or_false

) external onlyOwner {

    // Buyer Protection - Blacklisting can only be done before
launch
    if (true_or_false){require(LaunchTime == 0, "E08");}
    _isBlacklisted[Wallet] = true_or_false;

}
```

7. Trading calmdown

```
function Launch_Settings_02__Set_Launch_Limits(

    uint256 Launch_Buy_Delay_Seconds,
    uint256 Launch_Transaction_Limit_TOKENS,
    uint256 Launch_Phase_Length_Minutes

) external onlyOwner {

    max_Tran_Launch = Launch_Transaction_Limit_TOKENS * 10 **
_decimals;
    Launch_Buy_Delay = Launch_Buy_Delay_Seconds;
    Launch_Length = Launch_Phase_Length_Minutes * 60;
}
```

The owner can set calmdown trade but **no limit** in time

READ CONTRACT (ONLY NEED TO KNOW)

1. Token Information

show all token information uint256

(Shows Contract Information)

2. TradeOpen

false bool

(Shows trade open status)

3. Wallet_BNB

0x44ea309d694f6f94f2febd1c0f0e63c7b5ee2263 address

(Function for read wallet bnb receiver)

4. Wallet_Burn

0x00dead address

(Function for read burn address)

5. Wallet_Liquidity

0x11a7be8ae0ff813b6632c4b193887b57f1af11f7 uint256

(Function for read lp owner)

6. name

CRISTOMONEDAS 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)

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 is Official Audit Partner Of PinkSale

BlockSAFU Token Scanner

0xe748d5c3a572f14d07d4a16e0ef38a96f891156

Scan

There is no liquidity available for this contract.

BlockSAFU Token Scanner Score:

75

Score

Token Information

Indicator	Value
Token Name	CRISTOMONEDAS
Token Symbol	CSM
Total Supply	7,000,000
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	✔ Not detected
Proxy	✔ Not detected
Whitelisted	✔ Not detected
Anti Whale	❌ Detected
Trading Cooldown	❌ Detected
Transfer Pausable	✔ Not detected
Cannot Sell All	✔ Not detected

Rug Pull Safety

Indicator	Value
Hidden Owner	✔ Not detected
Creator Address	0xeb920d80...0fc 🔗
Creator Balance	0 CSM
Creator Percent	0%
Owner Address	0x705efc57...b37 🔗
Owner Balance	7,000,000 CSM
Owner Percent	100%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	✔ Not detected

WEBSITE REVIEW



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

Web-Tech stack: Wordpress, cloudflare

Domain .com (enom) - Tracked by whois

First Contentful Paint:	1.3s
Fully Loaded Time	4.2s
Performance	95%
Accessibility	96%
Best Practices	92%
SEO	100%

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 on dex

- TOP 5 Holder.

will be updated after listing on dex

- The team is no KYC By Blocksafu

HONEYPOT REVIEW

- Ability to sell.

- The owner is not able to pause the contract.

- The owner can't set fees over 20%

- The owner can set blacklist

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.