Coinsult

# Advanced Manual Smart Contract Audit

**Project:** BSCFlip
**Website:** https://www.bscflip.com

● **Low-risk**
4 low-risk code issues found

● **Medium-risk**
0 medium-risk code issues found

● **High-risk**
0 high-risk code issues found

**Contract address**
0xC1e64a579E7399bF90E05cD3EE5958840a79bFdF (relaunch)

## Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult checks the contract for coding issues, we do not guarantee the use case of the code. We do not check the logic of the functions and if they are used for what they were intended to be used.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

## Tokenomics

| Rank | Address | Quantity (Token) | Percentage |
|---|---|---|---|
| 1 | 0x2c3de508c770a44f2902259f1800aa798f25ee06 | 10,000,000 | 100.0000% |

## Source code

Coinsult was commissioned by BSCFlip to perform an audit based on the following smart contract:

https://bscscan.com/address/0xC1e64a579E7399bF90E05cD3EE5958840a79bFdF#code

# Manual Code Review

● **Low-risk**

4 low-risk code issues found.
Could be fixed, will not bring problems.

- Functions that send Ether to arbitrary destinations
  Unprotected call to a function sending Ether to an arbitrary address.

```
        addLiquidity(otherHalfLiq, liqFunds);
        (bool sent, bytes memory data) = _devAddress.call{value: 
devFunds}("");
        (bool sent1, bytes memory data1) = 
_marketingAddress.call{value: marketingFunds}("");
        (bool sent2, bytes memory data2) = _teamAddress.call{value: 
teamFunds}("");
        require(sent && sent1 && sent2, "Failed to send BNB");
        try distributor.deposit{value: rewardsFunds}() {} catch {}
```

- Contract contains Reentrancy vulnerabilities:
  _transferWithTaxes(address,address,uint256)

```
    function _transferWithTaxes(address from, address to, uint256 amount) private {
        // Sell tokens for funding
        if(
            !inSwapAndLiquify &&                                // Swap is not locked
            balanceOf(address(this)) >= _liquifyThreshhold &&   // liquifyThreshhold is
reached
            from != pair                                        // Not from liq pool (can't
sell during a buy)
        ) {
            swapCollectedFeesForFunding();
        }

        // Send fees to contract if necessary
        uint8 txType = 0;
        if (from == pair) txType = BUYTX;
        if (to == pair) txType = SELLTX;
        if(
            txType != 0 &&
            !(_isExcludedFromFees[from] || _isExcludedFromFees[to])
            && ((txType == BUYTX && _totalBuyTaxes > 0)
            || (txType == SELLTX && _totalSellTaxes > 0))
        ) {
            uint256 feesToContract = calculateTotalFees(amount, txType);

            if (feesToContract > 0) {
                amount = amount.sub(feesToContract);
                _transfer(from, address(this), feesToContract);
            }
        }

        _transfer(from, to, amount);
    }
```

- Costly operations inside a loop
  Use a local variable to hold the loop computation result.

```
        while(gasUsed < gas && iterations < shareholderCount) {
            if(currentIndex >= shareholderCount){
                currentIndex = 0;
            }

            if(shouldDistribute(shareholders[currentIndex])){
                distributeDividend(shareholders[currentIndex]);
            }

            gasUsed = gasUsed.add(gasLeft.sub(gasleft()));
            gasLeft = gasleft();
            currentIndex++;
            iterations++;
        }
```

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

```
    function addLiquidity(uint256 tokenAmount, uint256 ethAmount)
private {
        _approve(address(this), address(router), tokenAmount);

        router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            address(0),
            block.timestamp
        );
    }
```

## ● Medium-risk

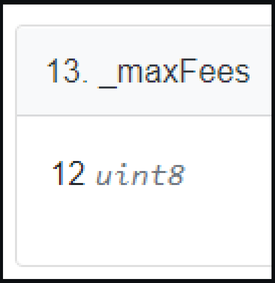
0 medium-risk code issues found.
Should be fixed, could bring problems.

## ● High-risk

0 high-risk code issues found
Must be fixed, and will bring problems.

## Extra notes by the team

🟢 Owner can change the buy and sell fees up to maxFees (constant)


13. _maxFees

12 *uint8*

```
    function setBuyFees(uint8 newDevBuyFee, uint8 newRewardsBuyFee, uint8
newMarketingBuyFee, uint8 newTeamBuyFee, uint8 newLiqBuyFee) external onlyOwner {
        uint8 newTotalBuyFees = newDevBuyFee + newRewardsBuyFee + newMarketingBuyFee +
newTeamBuyFee + newLiqBuyFee;
        require(!inSwapAndLiquify, "inSwapAndLiquify");
        require(newDevBuyFee <= _maxDevFee, "Cannot set dev fee higher than max");
        require(newTotalBuyFees <= _maxFees, "Cannot set total buy fees higher than max");

        _buyTaxes = Taxes({ devFee: newDevBuyFee, rewardsFee: newRewardsBuyFee,
marketingFee: newMarketingBuyFee,
            teamFee: newTeamBuyFee, liqFee: newLiqBuyFee });
        _totalBuyTaxes = newTotalBuyFees;
    }
```

```
    function setSellFees(uint8 newDevSellFee, uint8 newRewardsSellFee, uint8
newMarketingSellFee, uint8 newTeamSellFee, uint8 newLiqSellFee) external onlyOwner {
        uint8 newTotalSellFees = newDevSellFee + newRewardsSellFee + newMarketingSellFee +
newTeamSellFee + newLiqSellFee;
        require(!inSwapAndLiquify, "inSwapAndLiquify");
        require(newDevSellFee <= _maxDevFee, "Cannot set dev fee higher than max");
        require(newTotalSellFees <= _maxFees, "Cannot set total sell fees higher than
max");

        _sellTaxes = Taxes({ devFee: newDevSellFee, rewardsFee: newRewardsSellFee,
marketingFee: newMarketingSellFee,
            teamFee: newTeamSellFee, liqFee: newLiqSellFee });
        _totalSellTaxes = newTotalSellFees;
    }
```

- Owner can exclude from fees

- Owner can set max transaction amount with limit of 0.5% total supply

```
    // Set the max transaction percentage in increments of 0.1%.
    function setMaxTxPercentage(uint256 newMaxTxPercentage) external
onlyOwner {
        uint256 newMaxTx =
_totalSupply.mul(newMaxTxPercentage).div(1000);

        require(newMaxTx != _maxTx, "Cannot set new max transaction to
the same value as current max transaction");
        require(newMaxTx >= _totalSupply.mul(5).div(1000), "Cannot set
max transaction lower than 0.5 percent");

        _maxTx = newMaxTx;
    }
```

- Owner can exclude from max transaction amount

- Owner can set max balance amount with limit of 2% total supply

```
    function setMaxBalancePercentage(uint256 newMaxBalancePercentage)
external onlyOwner() {
        uint256 newMaxBalance =
_totalSupply.mul(newMaxBalancePercentage).div(100);

        require(newMaxBalance != _maxBalance, "Cannot set new max
balance to the same value as current max balance");
        require(newMaxBalance >= _totalSupply.mul(2).div(100), "Cannot
set max balance lower than 2 percent");

        _maxBalance = newMaxBalance;
    }
```

- Owner can exclude from max balance amount

- The ownership of the contract isn't renounced

● The owner can withdraw tokens from the contract

```
    // If you need to withdraw tokens that have been sent to the
contract
    function withdrawToken(address _tokenContract, uint256 _amount)
external onlyOwner {
        IERC20 tokenContract = IERC20(_tokenContract);

        // transfer the token from address of this contract
        // to address of the user (executing the withdrawToken()
function)
        tokenContract.transfer(msg.sender, _amount);
    }
```

# Contract Snapshot

```
contract BSCFlip is Context, Ownable, Taxable {
    using SafeMath for uint256;
    using Address for address;

    string private _Bname = "BSC Flip";
    string private _Bsymbol = "BSCF";
    // 9 Decimals
    uint8 private _Bdecimals = 18;
    // 10M Supply
    uint256 private _BtotalSupply = 10**7 * 10**_Bdecimals;
    // 2% Max Wallet
    uint256 private _BmaxBalance = _BtotalSupply.mul(2).div(100);
    // 0.5% Max Transaction
    uint256 private _BmaxTx = _BtotalSupply.mul(5).div(1000);
    // 12% Max Fees
    uint8 private _BmaxFees = 12;
    // 2% Max Dev Fee
    uint8 private _BmaxDevFee = 3;
    // Contract sell at 30k tokens
    uint256 private _BliquifyThreshhold = 3 * 10**4 * 10**_Bdecimals;
    TokenDistribution private _BtokenDistribution =
        TokenDistribution({ totalSupply: _BtotalSupply, decimals:
_Bdecimals, maxBalance: _BmaxBalance, maxTx: _BmaxTx });

    address payable _BdevAddress =
payable(address(0x2c3DE508c770a44F2902259f1800aA798f25ee06));
    address payable _BmarketingAddress =
payable(address(0x7C29E5F9F7DB90E830bf42EEAc36ffBaE30A67cB));
    address payable _BteamAddress =
payable(address(0x3252950D0ad561BF2E3689BA43C863456574ec6D));

    // Buy and sell fees will start at 99% to prevent bots/snipers at
launch,
    // but will not be allowed to be set this high ever again.
    constructor ()
    Taxable(_Bsymbol, _Bname, _BtokenDistribution, _BdevAddress,
_BmarketingAddress, _BteamAddress,
            Taxes({ devFee: 1, rewardsFee: 2, marketingFee: 32,
teamFee: 3, liqFee: 61 }),
```