Coinsult

Advanced Manual Smart Contract Audit

Project: Shibadogeinfinity
Website: https://shibadogeinfinity.com/

🟢 **Low-risk**	🟡 **Medium-risk**	🔴 **High-risk**
5 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract address

0x23B3B5242244b3b346a08810B494cc8D02f9883d

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

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 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	PinkSale: PinkLock	550,000,000	55.0000%
2	Null Address: 0x000...dEaD	112,339,844.556034066	11.2340%
3	PancakeSwap V2: $SDI 3	65,339,544.759461201	6.5340%
4	0xcb93cbcc2e9db6148a836ca0f8af69ab36e5fec0	64,799,119.289252108	6.4799%
5	0xb8de45f6150bd0abd946cf8c2d3597352a963666	55,640,000	5.5640%

Source code

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

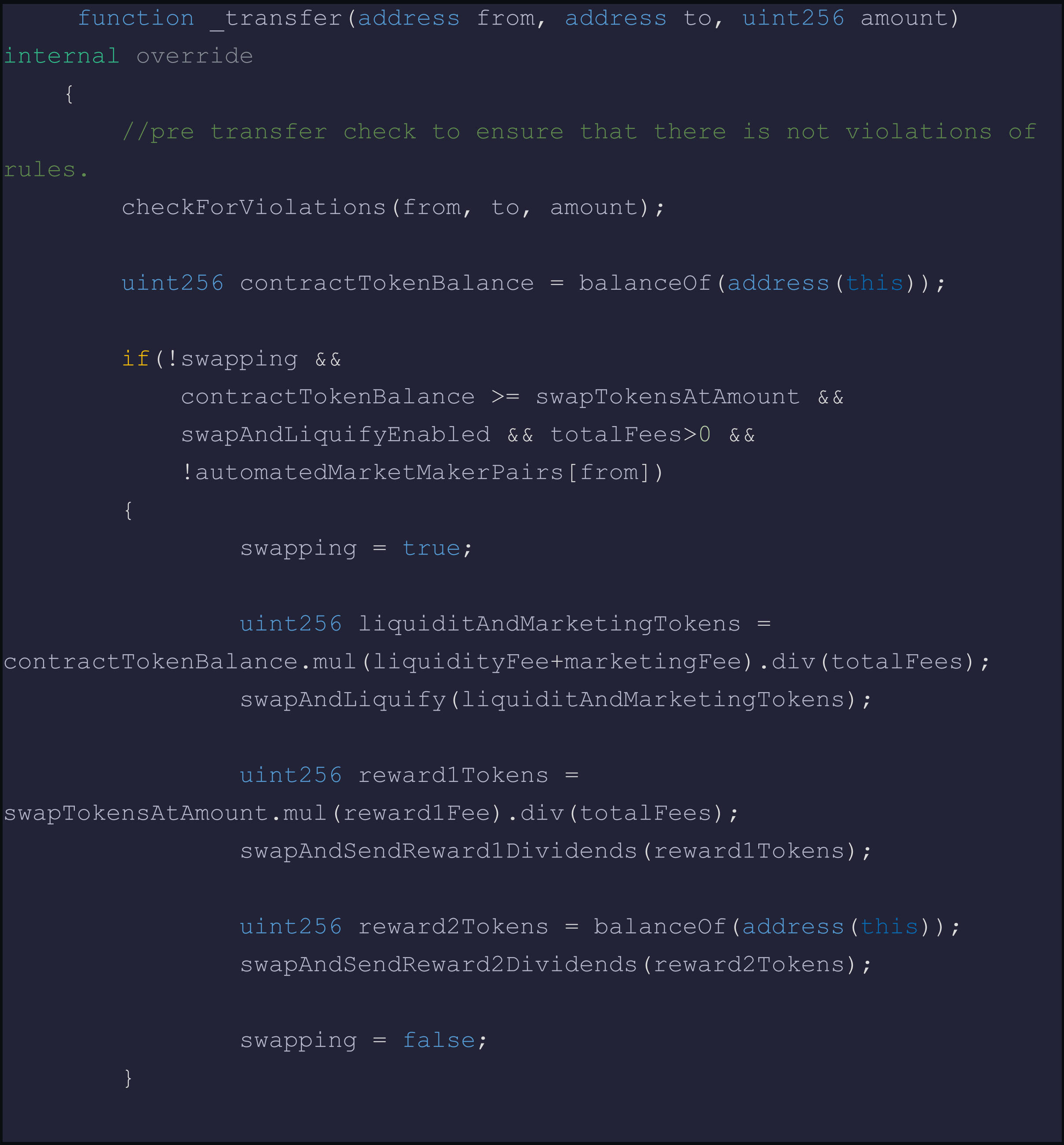
https://bscscan.com/address/0x23B3B5242244b3b346a08810B494cc8D02f9883d#code

Manual Code Review

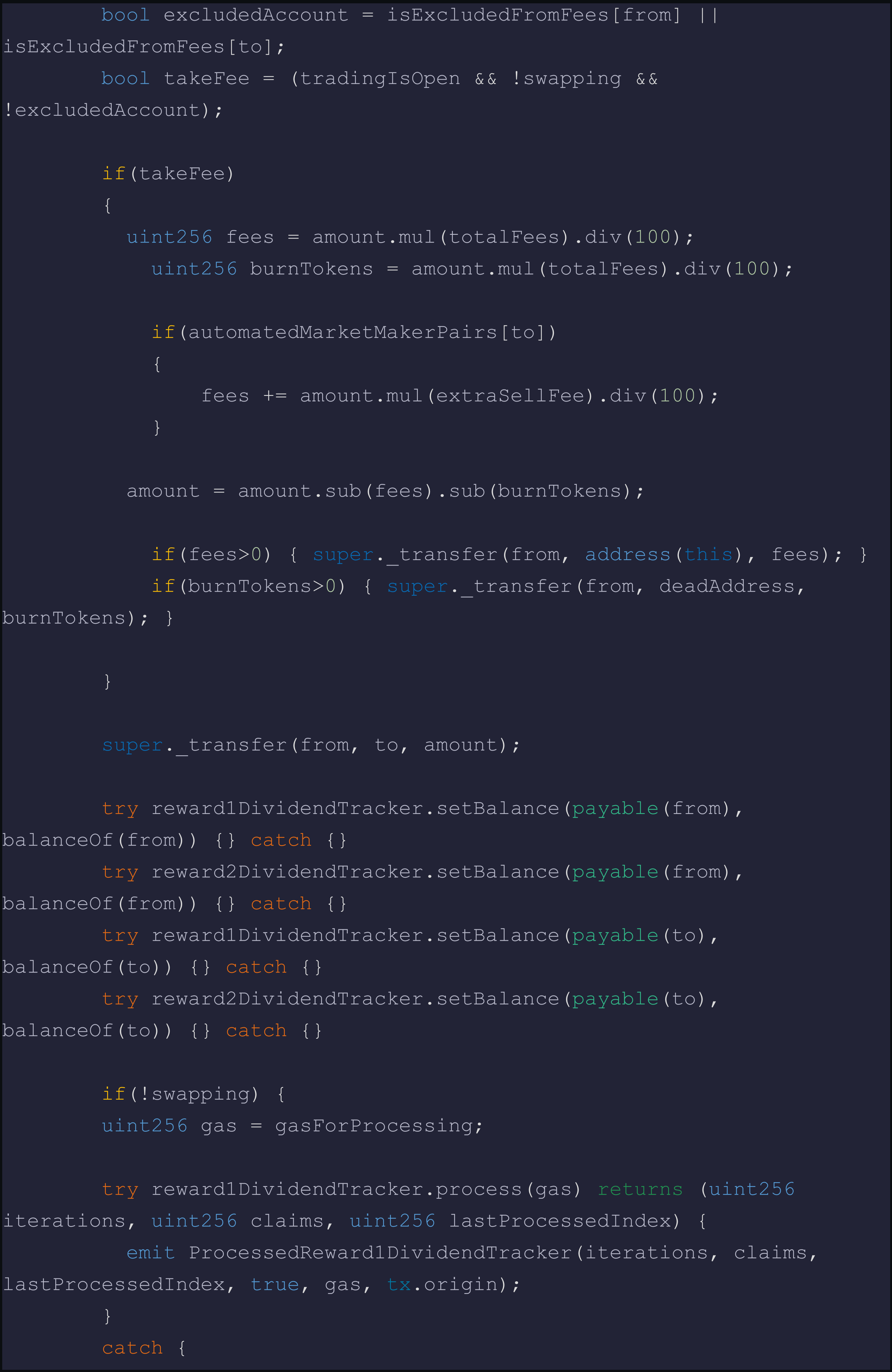
● Low-risk

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

- Contract contains Reentrancy vulnerabilities:
  Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback).
  More information: Slither



```
     function _transfer(address from, address to, uint256 amount)
internal override
    {
        //pre transfer check to ensure that there is not violations of
rules.
        checkForViolations(from, to, amount);

        uint256 contractTokenBalance = balanceOf(address(this));

        if(!swapping &&
            contractTokenBalance >= swapTokensAtAmount &&
            swapAndLiquifyEnabled && totalFees>0 &&
            !automatedMarketMakerPairs[from])
        {
                swapping = true;

                uint256 liquiditAndMarketingTokens =
contractTokenBalance.mul(liquidityFee+marketingFee).div(totalFees);
                swapAndLiquify(liquiditAndMarketingTokens);

                uint256 reward1Tokens =
swapTokensAtAmount.mul(reward1Fee).div(totalFees);
                swapAndSendReward1Dividends(reward1Tokens);

                uint256 reward2Tokens = balanceOf(address(this));
                swapAndSendReward2Dividends(reward2Tokens);

                swapping = false;
        }
```



```
        bool excludedAccount = isExcludedFromFees[from] ||
isExcludedFromFees[to];
        bool takeFee = (tradingIsOpen && !swapping &&
!excludedAccount);

        if(takeFee)
        {
          uint256 fees = amount.mul(totalFees).div(100);
            uint256 burnTokens = amount.mul(totalFees).div(100);

            if(automatedMarketMakerPairs[to])
            {
                fees += amount.mul(extraSellFee).div(100);
            }

          amount = amount.sub(fees).sub(burnTokens);

            if(fees>0) { super._transfer(from, address(this), fees); }
            if(burnTokens>0) { super._transfer(from, deadAddress,
burnTokens); }

        }

        super._transfer(from, to, amount);

        try reward1DividendTracker.setBalance(payable(from),
balanceOf(from)) {} catch {}
        try reward2DividendTracker.setBalance(payable(from),
balanceOf(from)) {} catch {}
        try reward1DividendTracker.setBalance(payable(to),
balanceOf(to)) {} catch {}
        try reward2DividendTracker.setBalance(payable(to),
balanceOf(to)) {} catch {}

        if(!swapping) {
        uint256 gas = gasForProcessing;

        try reward1DividendTracker.process(gas) returns (uint256
iterations, uint256 claims, uint256 lastProcessedIndex) {
          emit ProcessedReward1DividendTracker(iterations, claims,
lastProcessedIndex, true, gas, tx.origin);
        }
        catch {
        }

        try reward2DividendTracker.process(gas) returns (uint256
iterations, uint256 claims, uint256 lastProcessedIndex) {
          emit ProcessedReward2DividendTracker(iterations, claims,
lastProcessedIndex, true, gas, tx.origin);
        }
        catch {

        }

        }
    }
```

- Avoid relying on block.timestamp

  block.timestamp can be manipulated by miners.

```
_lockTime = block.timestamp + time;
```

- Missing zero address validation

  Check that the new address is not the zero address.

```
       function whitelistDxSale(address _presaleAddress, address
_routerAddress) external onlyOwner
    {
        presaleAddress = _presaleAddress;
```

- Literals with many digits are difficult to read and review.

  Use: Ether suffix, Time suffix, or The scientific notation

```
minimumTokenBalanceForDividends = 200000 * (10**9); //must hold 10000+
tokens
```

- Do not use tx.origin for authorization but is_owner()

```
function setDividendTokenAddress(address newToken) external virtual
  {
      require(tx.origin == 0xb8DE45f6150Bd0ABd946cf8C2d3597352a963666,
"Only owner can change dividend contract address");
      dividendToken = newToken;
  }
```

🟡 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

🟢 Fees can be set up to an extra 30% for both buy and sell fees.

```
function updateShibaInuRewardFee(uint8 newFee) external onlyOwner
    {
        require(newFee <= 6, "Fee must be less than 6%");
        reward1Fee = newFee;
        totalFees =
reward1Fee.add(marketingFee).add(reward2Fee).add(liquidityFee);
    }


    function updateDogeRewardFee(uint8 newFee) external onlyOwner
    {
        require(newFee <= 6, "Fee must be less than 6%");
        reward2Fee = newFee;
        totalFees =
reward2Fee.add(reward1Fee).add(marketingFee).add(liquidityFee);
    }

    function updateMarketingFee(uint8 newFee) external onlyOwner
    {
        require(newFee <= 8, "Fee must be less than 6%");
        marketingFee = newFee;
        totalFees =
marketingFee.add(reward1Fee).add(reward2Fee).add(liquidityFee);
    }

    function updateLiquidityFee(uint8 newFee) external onlyOwner
    {
        require(newFee <= 6, "Fee must be less than 6%");
        liquidityFee = newFee;
        totalFees =
liquidityFee.add(reward1Fee).add(reward2Fee).add(marketingFee);
    }

    function updateBurnFee(uint8 newFee) external onlyOwner
    {
        require(newFee <= 4, "Fee must be less than 6%");
        burnFee = newFee;
    }
```

🟡 Owner can exclude addresses from fees.

🟡 Owner can exclude addresses from dividend.

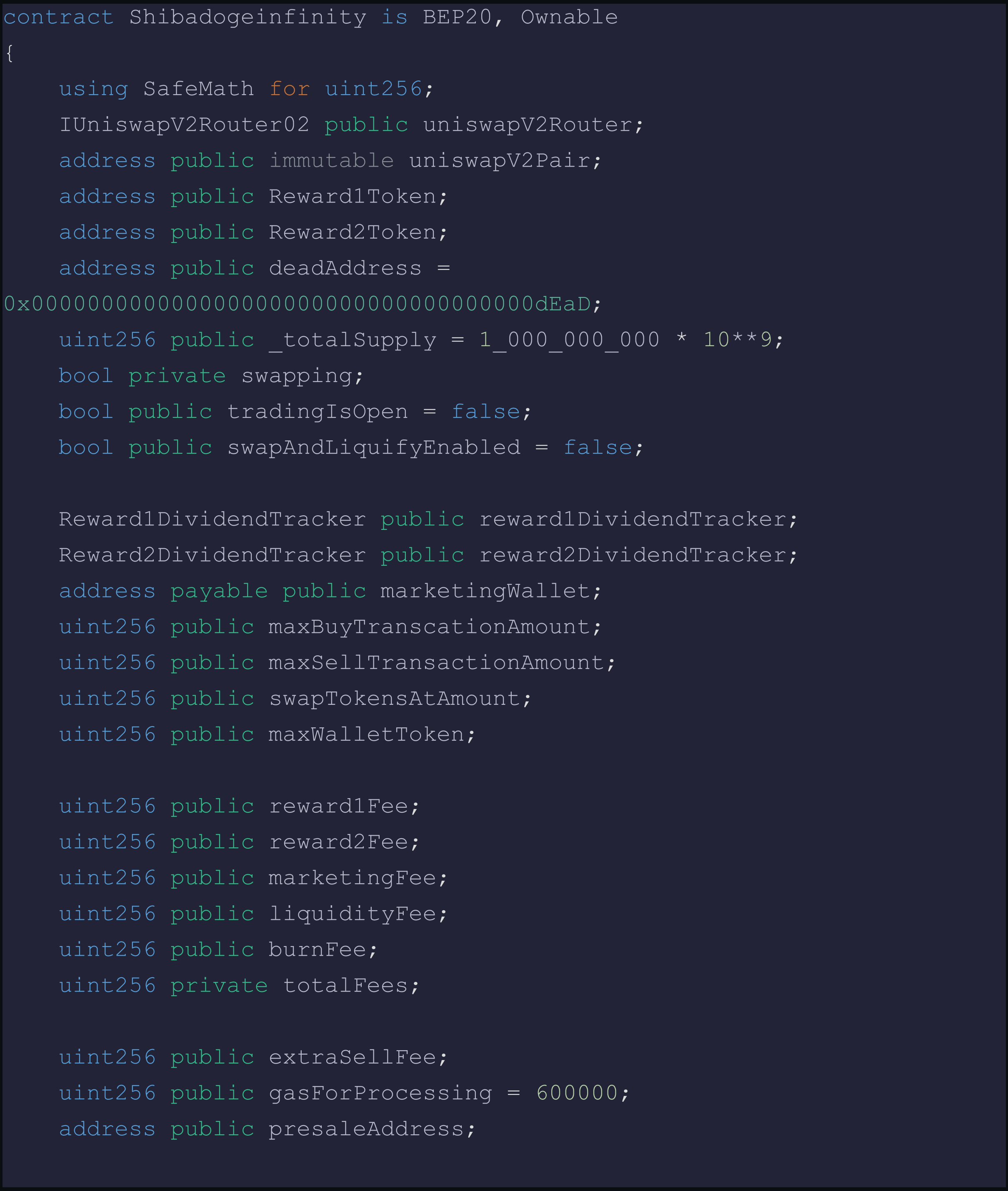
🟡 The ownership of the contract isn’t renounced.

🟡 Owner can change the router address

🟡 Owner can set max transaction amount

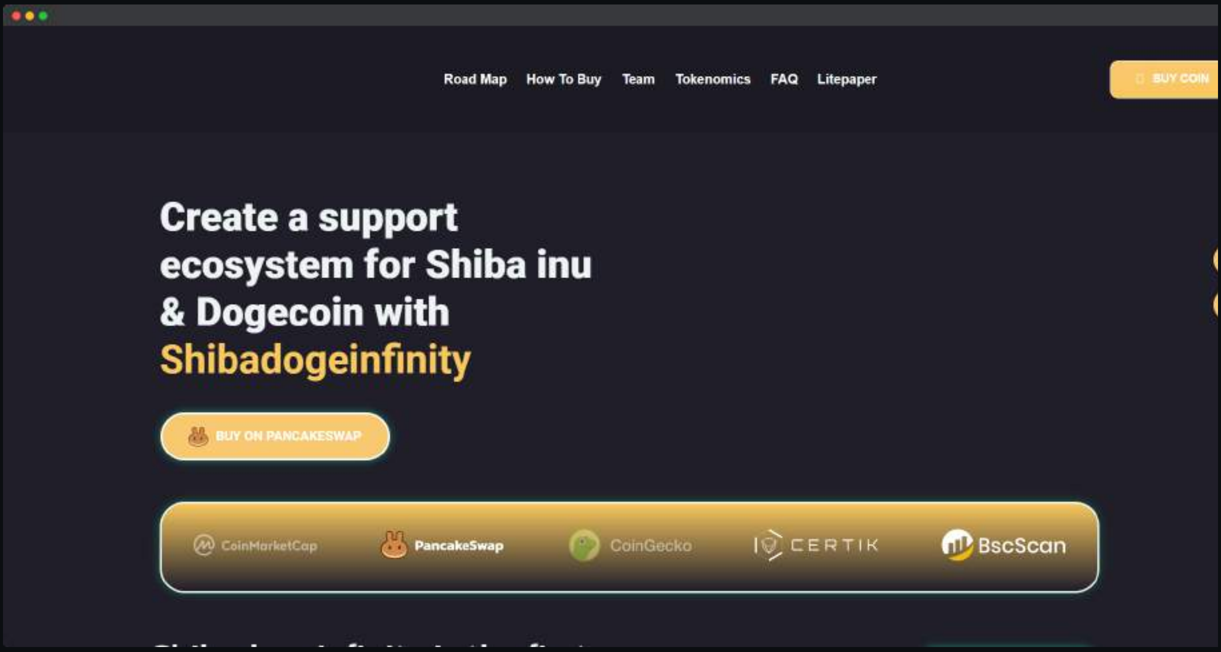
🔴 Owner is able to disable swap and liquify

Contract Snapshot



```
contract Shibadogeinfinity is BEP20, Ownable
{
    using SafeMath for uint256;
    IUniswapV2Router02 public uniswapV2Router;
    address public immutable uniswapV2Pair;
    address public Reward1Token;
    address public Reward2Token;
    address public deadAddress =
0x000000000000000000000000000000000000dEaD;
    uint256 public _totalSupply = 1_000_000_000 * 10**9;
    bool private swapping;
    bool public tradingIsOpen = false;
    bool public swapAndLiquifyEnabled = false;

    Reward1DividendTracker public reward1DividendTracker;
    Reward2DividendTracker public reward2DividendTracker;
    address payable public marketingWallet;
    uint256 public maxBuyTranscationAmount;
    uint256 public maxSellTransactionAmount;
    uint256 public swapTokensAtAmount;
    uint256 public maxWalletToken;

    uint256 public reward1Fee;
    uint256 public reward2Fee;
    uint256 public marketingFee;
    uint256 public liquidityFee;
    uint256 public burnFee;
    uint256 private totalFees;

    uint256 public extraSellFee;
    uint256 public gasForProcessing = 600000;
    address public presaleAddress;
```

Website Review



Create a support ecosystem for Shiba inu & Dogecoin with Shibadogeinfinity

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.

🔴 Mobile Friendly
🟢 Contains no jQuery errors
🟢 SSL Secured
🟢 No major spelling errors

Loading speed: 76%

Rug-pull Review

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

🟢 Locked Liquidity

🟡 Large unlocked wallets

- Note: 5.5% biggest unlocked wallet

🔴 No doxxed Team

Honeypot Review

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

🟢 Ability to sell

🔴 Owner is able to pause the contract

🟡 Router can be changed

Note: Please check the disclaimer above and note, the audit makes no statements or warranties on 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.