Coinsult

Advanced Manual Smart Contract Audit

Project: Dogecoin 2.0
Website: https://dogecoin.com/

Low-risk	Medium-risk	High-risk
6 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract address
0x35bB94258EB3fB42114dbdC0e29e00e5aBe0bc46

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	0xbe7a8020ffc741337f6938457c9fed8c5ac3c9f6	1,000,000,000,000,000	100.000%

Source code

Coinsult was commissioned by Dogecoin2.0 to perform an audit based on the following smart contract:

https://bscscan.com/address/0x35bB94258EB3fB42114dbdC0e29e00e5aBe0bc46#code

Manual Code Review

🟢 Low-risk

6 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 {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        require(!_isBlacklisted[from] && !_isBlacklisted[to], 'Blacklisted address');

        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if( canSwap &&
            !swapping &&
            !automatedMarketMakerPairs[from] &&
            from != owner() &&
            to != owner() &&
            swapAndLiquifyEnabled
        ) {
            swapping = true;
```

```
            if(AmountMarketingFee > 0)
swapAndSendToFee(AmountMarketingFee);
            if(AmountLiquidityFee > 0)
swapAndLiquify(AmountLiquidityFee);
            if(AmountTokenRewardsFee > 0)
swapAndSendDividends(AmountTokenRewardsFee);
            swapping = false;
        }


        bool takeFee = !swapping;

        // if any account belongs to _isExcludedFromFee account then
remove the fee
        if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
            takeFee = false;
        }

        if(takeFee) {
            uint256 fees;
            uint256 LFee; // Liquidity
            uint256 RFee; // Rewards
            uint256 MFee; // Marketing
            uint256 DFee; // Dead
            if(automatedMarketMakerPairs[from]){
                LFee = amount.mul(buyLiquidityFee).div(100);
                AmountLiquidityFee += LFee;
                RFee = amount.mul(buyTokenRewardsFee).div(100);
                AmountTokenRewardsFee += RFee;
                MFee = amount.mul(buyMarketingFee).div(100);
                AmountMarketingFee += MFee;
                DFee = amount.mul(buyDeadFee).div(100);
                fees = LFee.add(RFee).add(MFee).add(DFee);
            }
            if(automatedMarketMakerPairs[to]){
                LFee = amount.mul(sellLiquidityFee).div(100);
                AmountLiquidityFee += LFee;
                RFee = amount.mul(sellTokenRewardsFee).div(100);
                AmountTokenRewardsFee += RFee;
                MFee = amount.mul(sellMarketingFee).div(100);
                AmountMarketingFee += MFee;
                DFee = amount.mul(sellDeadFee).div(100);
                fees = LFee.add(RFee).add(MFee).add(DFee);
```

```
            }
            amount = amount.sub(fees);
            if(DFee > 0) super._transfer(from, deadWallet, DFee);
            super._transfer(from, address(this), fees.sub(DFee));
        }

        super._transfer(from, to, amount);

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

        if(!swapping) {
            uint256 gas = gasForProcessing;

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

            }
        }
    }
```

- Avoid relying on block.timestamp
  block.timestamp can be manipulated by miners.

```
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );
```

- Missing zero address validation

  Check that the new address is not the zero address.

```
        uniswapPair = _uniswapV2Pair;
    address serviceFeeReceiver_,
```

- Unchecked transfer

  More information: Slither

```
    function swapAndSendToFee(uint256 tokens) private  {
        uint256 initialCAKEBalance =
IERC20(rewardToken).balanceOf(address(this));
        swapTokensForCake(tokens);
        uint256 newBalance =
(IERC20(rewardToken).balanceOf(address(this))).sub(initialCAKEBalance);
        IERC20(rewardToken).transfer(_marketingWalletAddress,
newBalance);
        AmountMarketingFee = AmountMarketingFee - tokens;
    }
```

- External calls inside loop (succes = IERC20 (rewardToken).transfer()). if “succes” has a fallback function it will always be false and therefore will never use the transfer() function.
  More information: Slither

```
function _withdrawDividendOfUser(address payable user)
    internal
    returns (uint256)
    {
        uint256 _withdrawableDividend = withdrawableDividendOf(user);
        if (_withdrawableDividend > 0) {
            withdrawnDividends[user] = withdrawnDividends[user].add(
                _withdrawableDividend
            );
            emit DividendWithdrawn(user, _withdrawableDividend);
            bool success = IERC20(rewardToken).transfer(
                user,
                _withdrawableDividend
            );

            if (!success) {
                withdrawnDividends[user] =
withdrawnDividends[user].sub(
                    _withdrawableDividend
                );
                return 0;
            }

            return _withdrawableDividend;
        }

        return 0;
    }
```

- Literals with many digits are difficult to read and review.
  Use: Ether suffix, Time suffix, or The scientific notation

```
require(newValue >= 200000 && newValue <= 500000, "GasForProcessing
must be between 200,000 and 500,000");
```

🟡 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 exclude addresses from dividends.

🟡 Owner can exclude addresses from fees.

🟡 The ownership of the contract isn't renounced.

🟡 Owner can change the router address

🟢 Fees can be set up to 25% for both buy and sell fees.

```
function setBuyTaxes(uint256 liquidity, uint256 rewardsFee, uint256
marketingFee, uint256 deadFee) external onlyOwner {

require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) <= 25,
"Total buy fee is over 25%");
        buyTokenRewardsFee = rewardsFee;
        buyLiquidityFee = liquidity;
        buyMarketingFee = marketingFee;
        buyDeadFee = deadFee;

    }
function setSelTaxes(uint256 liquidity, uint256 rewardsFee, uint256
marketingFee, uint256 deadFee) external onlyOwner {

require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) <= 25,
"Total sel fee is over 25%");
        sellTokenRewardsFee = rewardsFee;
        sellLiquidityFee = liquidity;
        sellMarketingFee = marketingFee;
        sellDeadFee = deadFee;
    }
```

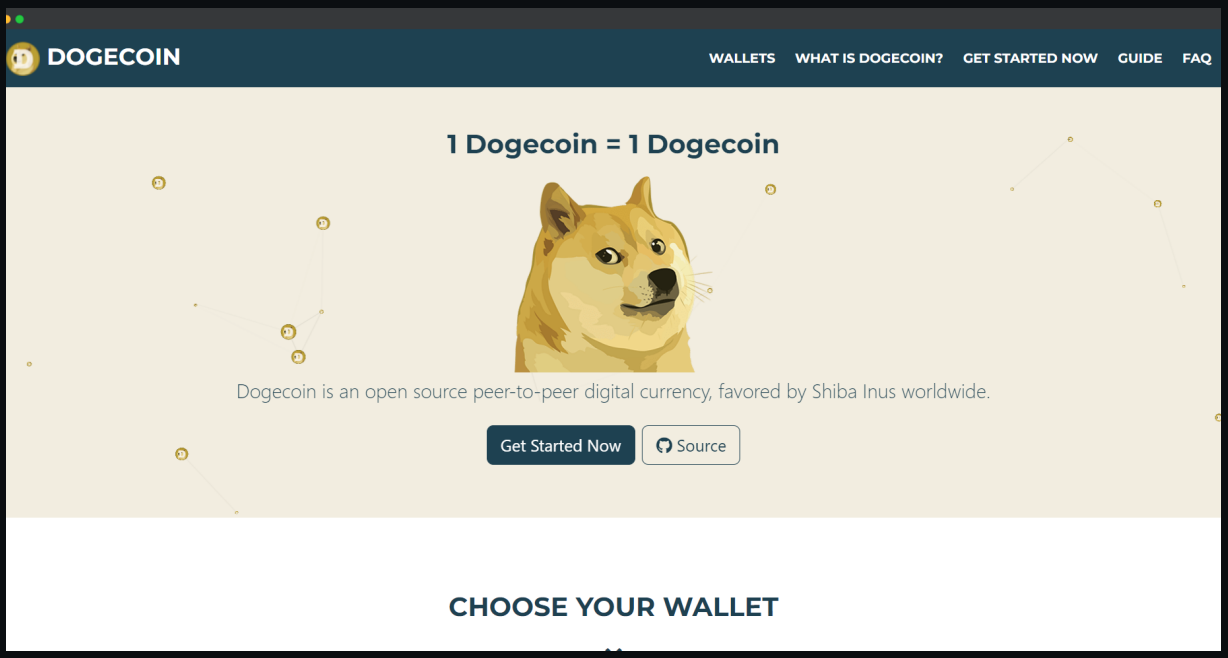
🔴 Owner is able to disable swap and liquify

🔴 Contract uses multiple external contracts which are not audited by coinsult. (BABYTOKENDividendTracker, BABYTOKEN, ERC20Upgradeable)

Contract Snapshot

```
contract ERC20 is Context, IERC20, IERC20Metadata {
    using SafeMath for uint256;

    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private
_allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    /**
     * @dev Sets the values for {name} and {symbol}.
     *
     * The default value of {decimals} is 18. To select a different
value for
     * {decimals} you should overload it.
     *
     * All two of these values are immutable: they can only be set once
during
     * construction.
     */
    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }
```

Website Review



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: 86%

Rug-pull Review

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

🟢 Locked Liquidity (no liquidity yet)

🟡 Large unlocked wallets

- Note: Tokens not distributed yet

🔴 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.