Coinsult

Advanced Manual Smart Contract Audit

Project: Gemie Inu
Website: https://gemieinu.finance

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

Contract address
0x0E3e2AB2743fbB0270b4b6658935aAb14a84b6f1

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

Total Supply: 10,000,000,000,000,000
Total Holders: 1
Top 10 holders:

Rank	Address	Quantity (Token)	Percentage
1	0xa268df4926fe0aaeb13ceb7ee9a579dc8b865234	10,000,000,000,000,000	100.0000%

The top 100 holders collectively own 100.00% (10,000,000,000,000,000.00 Tokens) of Gemie Inu

Note: This is a snapshot of when the audit was performed.

Source code

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

https://bscscan.com/address/0x0E3e2AB2743fbB0270b4b6658935aAb14a84b6f1#code

Manual Code Review

🟢 Low-risk

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

- Contract contains Reentrancy vulnerabilities: _transfer(address,address,uint256)
  Additional information: State variables written after the call(s), apply the check-effects-interactions pattern.

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) private {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");
        require(!blackList[from] && !blackList[to]);
        if(from != owner() && to != owner())
            require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
        // is the token balance of this contract address over the min number of
        // tokens that we need to initiate a swap + liquidity lock?
        // also, don't get caught in a circular liquidity event.
        // also, don't swap & liquify if sender is uniswap pair.
        uint256 contractTokenBalance = balanceOf(address(this));

        if(contractTokenBalance >= _maxTxAmount)
        {
            contractTokenBalance = _maxTxAmount;
        }

        bool overMinTokenBalance = contractTokenBalance >= numTokensSellToAddToLiquidity;
        if (
            overMinTokenBalance &&
            !inSwapAndLiquify &&
            from != uniswapV2Pair &&
            swapAndLiquifyEnabled
        ) {
            contractTokenBalance = numTokensSellToAddToLiquidity;
            //add liquidity
            swapAndLiquify(contractTokenBalance);
        }

        //indicates if fee should be deducted from transfer
        bool takeFee = true;

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

        //transfer amount, it will take tax, burn, liquidity fee
        _tokenTransfer(from,to,amount,takeFee);
    }
```

- Solidity defines a naming convention that should be followed.
  Follow the Solidity naming convention. (mixedCase)

```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (#596) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (#597) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (#614) is not in mixedCase
Function IUniswapV2Router01.WETH() (#634) is not in mixedCase
```

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

```
    uint256 private _tTotal = 10000000000 * 10**6 * 10**9;
    uint256 public _maxTxAmount = 2000000000 * 10**6 * 10**9;
    uint256 private numTokensSellToAddToLiquidity = 500000000 * 10**6 *
10**9;
```

🟡 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

🟢 Contract has hardcoded taxes: 9% buy tax and 8% sell tax

🟡 A lot of commented code is in the contract, this could be removed to increase the readability.

🟡 The ownership is not renounced.

🟡 `SafeMath` is generally not needed starting with Solidity 0.8, since the compiler now has built in overflow checking.

🔴 Owner can blacklist

Contract Snapshot

```
contract GemieInu is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;
    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private
_allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _isExcluded;
    mapping (address => bool) private blackList;
    address[] private _excluded;
    address private _charityWalletAddress =
0xC54826520DEaBEaC5B1109982150ab04b7A8c625;
    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 10000000000 * 10**6 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;
    string private _name = "Gemie Inu";
    string private _symbol = "GemieInu";
    uint8 private _decimals = 9;
    uint256 public _taxFee = 3;
    uint256 private _previousTaxFee = _taxFee;
    uint256 public _charityFee = 3;
    uint256 private _previousCharityFee = _charityFee;
    uint256 public _liquidityFee = 3;
    uint256 private _previousLiquidityFee = _liquidityFee;
    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;
    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = true;
    uint256 public _maxTxAmount = 2000000000 * 10**6 * 10**9;
    uint256 private numTokensSellToAddToLiquidity = 500000000 * 10**6 *
10**9;
    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyEnabledUpdated(bool enabled);
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

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: 87%

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

🟡 Not doxxed yet (will do KYC with PinkSale)

Honeypot Review

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

🟢 Ability to sell

🟢 Owner is not able to pause the contract

- Note: Owner can blacklist

🟢 Correct router hard coded in the contract

- 0x10ED43C718714eb63d5aA57B78B54704E256024E

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.