Coinsult

Advanced Manual Smart Contract Audit

Project: Sadaharu Inu
Website: https://www.sadaharu.io

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

Contract address
0x4Cb2bbE560cEdCA9c4E8dC7d38243444AEf03A22

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.

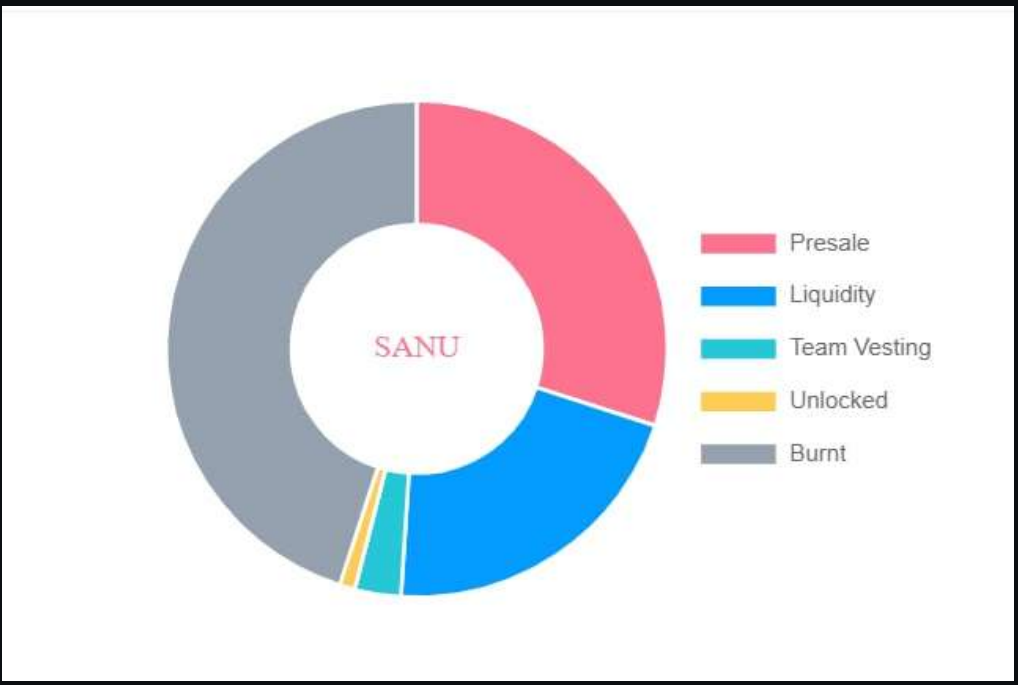
Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

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	0x9c3890b1a4d75791f3a9dc61f845f639bc9f7559	54,600,000,000	54.6000%
2	Null Address: 0x000...dEaD	45,000,000,000	45.0000%
3	0xca81dd937cdf995115019faa05ec5ee0ada21628	400,000,000	0.4000%



Source code

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

https://bscscan.com/address/0x4Cb2bbE560cEdCA9c4E8dC7d38243444AEf03A22#code

Manual Code Review

🟢 Low-risk

4 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
    ) 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");

        // 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));

        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);
    }
```

- Contradiction in the contract.
  Fix the incorrect comparison by changing the value type or the comparison. More information: Slither

```
require(taxFeeBps_ >= 0, "Invalid tax fee");
        require(liquidityFeeBps_ >= 0, "Invalid liquidity fee");
        require(charityFeeBps_ >= 0, "Invalid charity fee");
```

- Missing zero address validation
  Check that the new address is not the zero address.

```
    _charityAddress = charityAddress_;
```

- Costly operations inside a loop might waste gas, so optimizations are justified.
  Use a local variable to hold the loop computation result.

```
    function includeInReward(address account) external onlyOwner {
        require(_isExcluded[account], "Account is already excluded");
        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;
            }
        }
    }
```

🟡 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 fees.

🟡 The ownership of the contract isn't renounced.

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

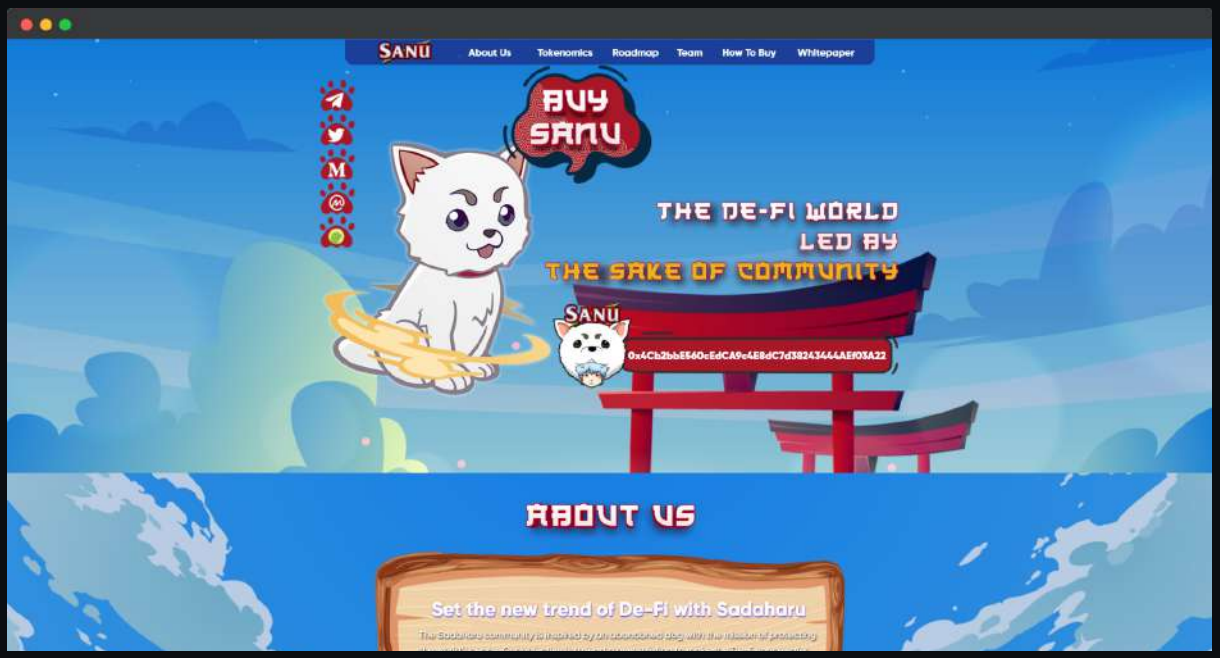
```
function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
        _taxFee = taxFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,
            "Total fee is over 25%"
        );
    }

    function setLiquidityFeePercent(uint256 liquidityFeeBps)
        external
        onlyOwner
    {
        _liquidityFee = liquidityFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= 10**4 / 4,
            "Total fee is over 25%"
        );
    }
```

🔴 Owner is able to disable swap and liquify

Contract Snapshot

```
contract LiquidityGeneratorToken is IERC20, Ownable, BaseToken {
    using SafeMath for uint256;
    using Address for address;

    uint256 public constant VERSION = 1;

    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;
    address[] private _excluded;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal;
    uint256 private _rTotal;
    uint256 private _tFeeTotal;

    string private _name;
    string private _symbol;
    uint8 private _decimals;

    uint256 public _taxFee;
    uint256 private _previousTaxFee = _taxFee;

    uint256 public _liquidityFee;
    uint256 private _previousLiquidityFee = _liquidityFee;

    uint256 public _charityFee;
    uint256 private _previousCharityFee = _charityFee;

    IUniswapV2Router02 public uniswapV2Router;
    address public uniswapV2Pair;
    address public _charityAddress;

    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled;
```

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)

🟢 Locked 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 cannot 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.