The Rare Antiquities Token

Smart Contract on ERC20

Contents

1. Project Overview

The Rare Antiquities Token is an ERC20 token that uses Lossless protocol and Biconomy Gasless protocol.

Lossless protocol provides an extra security layer in order to prevent fraudulent token movement and recover tokens that have been taken fraudulently (https://lossless-cash.gitbook.io/lossless/technical-reference/hack-mitigation-protocol).

Biconomy protocol provides a means for utilizing a gas bank to provide subsided gas transactions of the decentralized exchange (https://docs.biconomy.io/products/enable-gasless-transactions).

The Rare Antiquities Token also operates with taxes on all buys and sells which accumulate in WETH at a Marketing, Gas and Antiquities wallet address.

2. Functional Requirements

a. Roles

- **Contract Admin**
  Controls all contract functions such as enable trading, bot blacklists, fees, max tx, fee wallet deposit addresses.
- **User**
  Can buy and sell the token on the decentralised exchange.

b. Features

- Subsidise gas for transactions (User)
- Report fraudulent transactions (User)
- Change fee amounts for marketing, antiquities, gas. (Admin)
- Change fee deposit wallets. (Admin)
- Change max tx. (Admin)
- Blacklist bot wallet addresses. (Admin)

c. Use Cases

1. The admin changes the fees for the marketing, antiquities, and gas amounts. The fees are then taken as WETH from all transactions and deposited in the corresponding wallet address.
2. Admin can add a bot wallet address to the blacklist so that they are unable to make any transactions on the token.

3. Technical Requirements

a. Contract Information

TheRareAntiquitiesToken.sol

A token smart contract with a supply that can be traded on a decentralized exchange where liquidity is added.

- **marketingWallet :** marketing wallet address
- **antiquitiesWallet:** antiquities wallet address
- **gasWallet:** gas wallet address
- **function mul(uint256 a, uint256 b) internal pure returns (uint256) { :** Gas optimization
- **function renounceOwnership() public virtual onlyOwner {**
  emit OwnershipTransferred(_owner, address(0));
  _owner = address(0); : Renounce contract ownership
- **function transferOwnership(address newOwner) public virtual onlyOwner {**
  require(newOwner != address(0), "Ownable: new owner is the zero address");
  emit OwnershipTransferred(_owner, newOwner);
  _owner = newOwner;
  }: Transfer ownership
- **marketingFee :** marketing wallet fee
- **antiquitiesFee:** antiquities wallet fee
- **gasFee:** gas wallet fee
- **uint256 public _maxTxAmount = 500000000000 * 10**9; :** total supply by default, can be changed at will
- **uint256 public _maxWallet = 5000000000 * 10**9; :** 1% max wallet by default, can be changed at will
- **IRARESwapPair(rareSwapPair).setBaseToken(WETH);**
  IRARESwapPair(rareSwapPair).updateTotalFee(_marketingFee + _antiquitiesFee);
  _approve(_msgSender(), address(rareSwapRouter), _tTotal); Set base token in the pair as WETH, which acts as the tax token
- **rareSwapPair = IRARESwapFactory(rareSwapRouter.factory())**
  .createPair(address(this), WETH); Create a dex pair for this new token