require('dotenv').config();

const { DefenderRelayProvider, DefenderRelaySigner } = require('@openzeppelin/defender-relay-client');

const { ethers } = require('ethers');

async function main() {

const credentials = {

apiKey: process.env.DEFENDER\_API\_KEY,

apiSecret: process.env.DEFENDER\_API\_SECRET,

};

const provider = new DefenderRelayProvider(credentials);

const signer = new DefenderRelaySigner(credentials, provider, { speed: 'fast' });

// --- CONFIGURATION ---

// Replace with your deployed ELMC token address

const elmcAddress = process.env.ELMC\_CONTRACT\_ADDRESS; // Must be set in .env

// Replace with the actual PancakeSwap Pair address (LP token address)

const pairAddress = process.env.PANCAKESWAP\_PAIR\_ADDRESS; // Must be set in .env

const burnAddress = "0x000000000000000000000000000000000000dEaD"; // Standard burn address

if (!elmcAddress || !pairAddress) {

console.error("ERROR: Please ensure ELMC\_CONTRACT\_ADDRESS and PANCAKESWAP\_PAIR\_ADDRESS are set in your .env file.");

process.exit(1);

}

// --- END CONFIGURATION ---

console.log("----------------------------------------------------");

console.log("🔥 Burning LP Tokens via Defender Relayer");

console.log("ELMC Contract Address:", elmcAddress);

console.log("PancakeSwap LP Token Address:", pairAddress);

console.log("Burn Address:", burnAddress);

console.log("----------------------------------------------------");

// ELMC Coin ABI (only the burnLPTokens function)

const elmcAbi = [

"function burnLPTokens(uint256 amount) external",

"function owner() view returns (address)" // To check owner

];

const elmc = new ethers.Contract(elmcAddress, elmcAbi, signer);

// Get the owner of the ELMC contract (which should be the Gnosis Safe after transfer)

const currentOwner = await elmc.owner();

console.log("Current ELMC Contract Owner:", currentOwner);

// Get LP token contract instance

const lpTokenAbi = [

"function balanceOf(address account) view returns (uint256)",

"function allowance(address owner, address spender) view returns (uint256)",

"function approve(address spender, uint256 amount) returns (bool)",

"function transferFrom(address from, address to, uint256 amount) returns (bool)"

];

const lpToken = new ethers.Contract(pairAddress, lpTokenAbi, signer);

// Check LP token balance of the ELMC contract owner (which is the Relayer's signer for this action)

const ownerLPBalance = await lpToken.balanceOf(currentOwner); // Check balance of the actual owner (Safe)

console.log("LP Token balance of ELMC contract owner (Safe):", ethers.formatEther(ownerLPBalance));

if (ownerLPBalance === 0n) {

console.log("No LP tokens found in the ELMC contract owner's wallet to burn.");

return;

}

// Approve the ELMCCoin contract to spend the LP tokens from the owner (Gnosis Safe)

// This step is crucial if the LP tokens are held by the Gnosis Safe.

const elmcContractAsSpender = elmcAddress; // The ELMCCoin contract itself needs approval to burn from owner's balance

const currentAllowance = await lpToken.allowance(currentOwner, elmcContractAsSpender);

if (currentAllowance < ownerLPBalance) {

console.log("Approving ELMCCoin contract to spend LP tokens from owner...");

const approveTx = await lpToken.approve(elmcContractAsSpender, ownerLPBalance);

console.log("Approval TX sent:", approveTx.hash);

await approveTx.wait();

console.log("✅ Approval confirmed.");

} else {

console.log("Sufficient approval already exists.");

}

// Call burnLPTokens on the ELMCCoin contract

console.log(`Calling burnLPTokens on ELMCCoin contract to burn ${ethers.formatEther(ownerLPBalance)} LP tokens...`);

const burnTx = await elmc.burnLPTokens(ownerLPBalance);

console.log("Burn TX sent:", burnTx.hash);

await burnTx.wait();

console.log(`✅ Successfully burned all LP tokens from owner's balance.`);

console.log("----------------------------------------------------");

}

main().catch((error) => {

console.error(error);

process.exitCode = 1;

});