RE-ENTRACY DATA SET :

EXAMPLE 1 :

/\*

\* @source: https://ethernaut.zeppelin.solutions/level/0xf70706db003e94cfe4b5e27ffd891d5c81b39488

\* @author: Alejandro Santander

\* @vulnerable\_at\_lines: 24

\*/

pragma solidity ^0.4.18;

contract Reentrance {

mapping(address => uint) public balances;

function donate(address \_to) public payable {

balances[\_to] += msg.value;

}

function balanceOf(address \_who) public view returns (uint balance) {

return balances[\_who];

}

function withdraw(uint \_amount) public {

if(balances[msg.sender] >= \_amount) {

// <yes> <report> REENTRANCY

if(msg.sender.call.value(\_amount)()) {

\_amount;

}

balances[msg.sender] -= \_amount;

}

}

function() public payable {}

}

EXAMPLE 2 :

/\*

\* @source: https://github.com/SmartContractSecurity/SWC-registry/blob/master/test\_cases/reentracy/modifier\_reentrancy.sol

\* @author: -

\* @vulnerable\_at\_lines: 15

\*/

pragma solidity ^0.4.24;

contract ModifierEntrancy {

mapping (address => uint) public tokenBalance;

string constant name = "Nu Token";

//If a contract has a zero balance and supports the token give them some token

// <yes> <report> REENTRANCY

function airDrop() hasNoBalance supportsToken public{

tokenBalance[msg.sender] += 20;

}

//Checks that the contract responds the way we want

modifier supportsToken() {

require(keccak256(abi.encodePacked("Nu Token")) == Bank(msg.sender).supportsToken());

\_;

}

//Checks that the caller has a zero balance

modifier hasNoBalance {

require(tokenBalance[msg.sender] == 0);

\_;

}

}

contract Bank{

function supportsToken() external pure returns(bytes32){

return(keccak256(abi.encodePacked("Nu Token")));

}

}

contract attack{ //An example of a contract that breaks the contract above.

bool hasBeenCalled;

function supportsToken() external returns(bytes32){

if(!hasBeenCalled){

hasBeenCalled = true;

ModifierEntrancy(msg.sender).airDrop();

}

return(keccak256(abi.encodePacked("Nu Token")));

}

function call(address token) public{

ModifierEntrancy(token).airDrop();

}

}

EXAMPLE 3 :

/\*

\* @source: https://consensys.github.io/smart-contract-best-practices/known\_attacks/

\* @author: consensys

\* @vulnerable\_at\_lines: 28

\*/

pragma solidity ^0.4.0;

contract Reentrancy\_bonus{

// INSECURE

mapping (address => uint) private userBalances;

mapping (address => bool) private claimedBonus;

mapping (address => uint) private rewardsForA;

function withdrawReward(address recipient) public {

uint amountToWithdraw = rewardsForA[recipient];

rewardsForA[recipient] = 0;

(bool success, ) = recipient.call.value(amountToWithdraw)("");

require(success);

}

function getFirstWithdrawalBonus(address recipient) public {

require(!claimedBonus[recipient]); // Each recipient should only be able to claim the bonus once

rewardsForA[recipient] += 100;

// <yes> <report> REENTRANCY

withdrawReward(recipient); // At this point, the caller will be able to execute getFirstWithdrawalBonus again.

claimedBonus[recipient] = true;

}

}

EXAMPLE 4 :

/\*

\* @source: https://github.com/ConsenSys/evm-analyzer-benchmark-suite

\* @author: Suhabe Bugrara

\* @vulnerable\_at\_lines: 18

\*/

pragma solidity ^0.4.19;

contract ReentrancyDAO {

mapping (address => uint) credit;

uint balance;

function withdrawAll() public {

uint oCredit = credit[msg.sender];

if (oCredit > 0) {

balance -= oCredit;

// <yes> <report> REENTRANCY

bool callResult = msg.sender.call.value(oCredit)();

require (callResult);

credit[msg.sender] = 0;

}

}

function deposit() public payable {

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

balance += msg.value;

}

}