```
[12/14/22]seed@VM:~/.../contract$ ls
Makefile  README.md  ReentrancyAttacker.sol  ReentrancyVictim.sol  solc-0.6.8
[12/14/22]seed@VM:~/.../contract$ solc-0.6.8 --overwrite --abi --bin -o . Reentr
ancyVictim.sol
Compiler run successful. Artifact(s) can be found in directory ..
```

Generate .abi and .bin files and we can see them.

```
[12/14/22]seed@VM:~/.../643_reentrancy$ python3 deploy_victim_contract.py
Deploying the victim contract ...
---------Deploying Contract ----------------
... Waiting for block
Transaction Hash: 0x9f79602533bd19ad2e0d0240658f5833619f3e828802a552fc89d99b4e
27c698
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0x38f842dc903758650
644cab9185c377788bdb5974af401ca15831336ba3f651d'), 'blockNumber': 313, 'contra
ctAddress': '0xDD091dC9A42CCf45d7Fc1F872794520fdF2d12EB', 'cumulativeGasUsed':
 282261, 'effectiveGasPrice': 1000000007, 'from': '0x63f73c74F8dc4AeD0d396bA12
13B033253a4eCa0', 'gasUsed': 282261, 'logs': [], 'logsBloom': HexBytes('0x0000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000'), 'status': 1, 'to': None, 'transact
ionHash': HexBytes('0x9f79602533bd19ad2e0d0240658f5833619f3e828802a552fc89d99b
4e27c698'), 'transactionIndex': 0, 'type': '0x2'})
Victim contract: 0xDD091dC9A42CCf45d7Fc1F872794520fdF2d12EB
```

Deploy victim contract and use the address for deploying attack victim and then funding the victim contract, withdrawing etc.

Open — fund_victim_contract.py — ~/Downloads/Blockchain_CSE643-20221110...cy-20221213T160507Z-001/643_reentrancy — Save

```
#!/bin/env python3

from web3 import Web3
import SEEDWeb3
import os

abi_file    = "/contract/ReentrancyVictim.abi"
victim_addr = '0xDD091dC9A42CCf45d7Fc1F872794520fdF2d12EB'

# Connect to our geth node
port = 8547
web3 = SEEDWeb3.connect_to_geth_poa('http://127.0.0.1:{}' .format(port))

# We use web3.eth.accounts[1] as the sender because it has more Ethers
sender_account = web3.eth.accounts[1]
web3.geth.personal.unlockAccount(sender_account, "admin")

# Deposit Ethers to the victim contract
# The attacker will steal them in the attack later
contract_abi  = SEEDWeb3.getFileContent(abi_file)
amount = 30  # the unit is ether
contract = web3.eth.contract(address=victim_addr, abi=contract_abi)
tx_hash  = contract.functions.deposit().transact({
                    'from':  sender_account,
```

Update the victim contract contact info and then enter the amount 30 that you want to transfer.

deploy_attack_contract.py | deploy_victim_contract.py | fund_victim_contract.py | withdraw_from_victim_contract.py

```
from web3 import Web3
import SEEDWeb3
import os

abi_file    = "contract/ReentrancyVictim.abi"
victim_addr = '0xDD091dC9A42CCf45d7Fc1F872794520fdF2d12EB'
filename = 'contract_address_victim.txt'


# Connect to our geth node
port = 8547
web3 = SEEDWeb3.connect_to_geth_poa('http://127.0.0.1:{}'.format(port))

# We use web3.eth.accounts[1] as the sender because it has more Ethers
sender_account = web3.eth.accounts[1]
web3.geth.personal.unlockAccount(sender_account, "admin")

# Deposit Ethers to the victim contract
# The attacker will steal them in the attack later
contract_abi  = SEEDWeb3.getFileContent(abi_file)
contract = web3.eth.contract(address=victim_addr, abi=contract_abi)
amount = 5
tx_hash = contract.functions.withdraw(Web3.toWei(amount, 'ether')).transact({
```

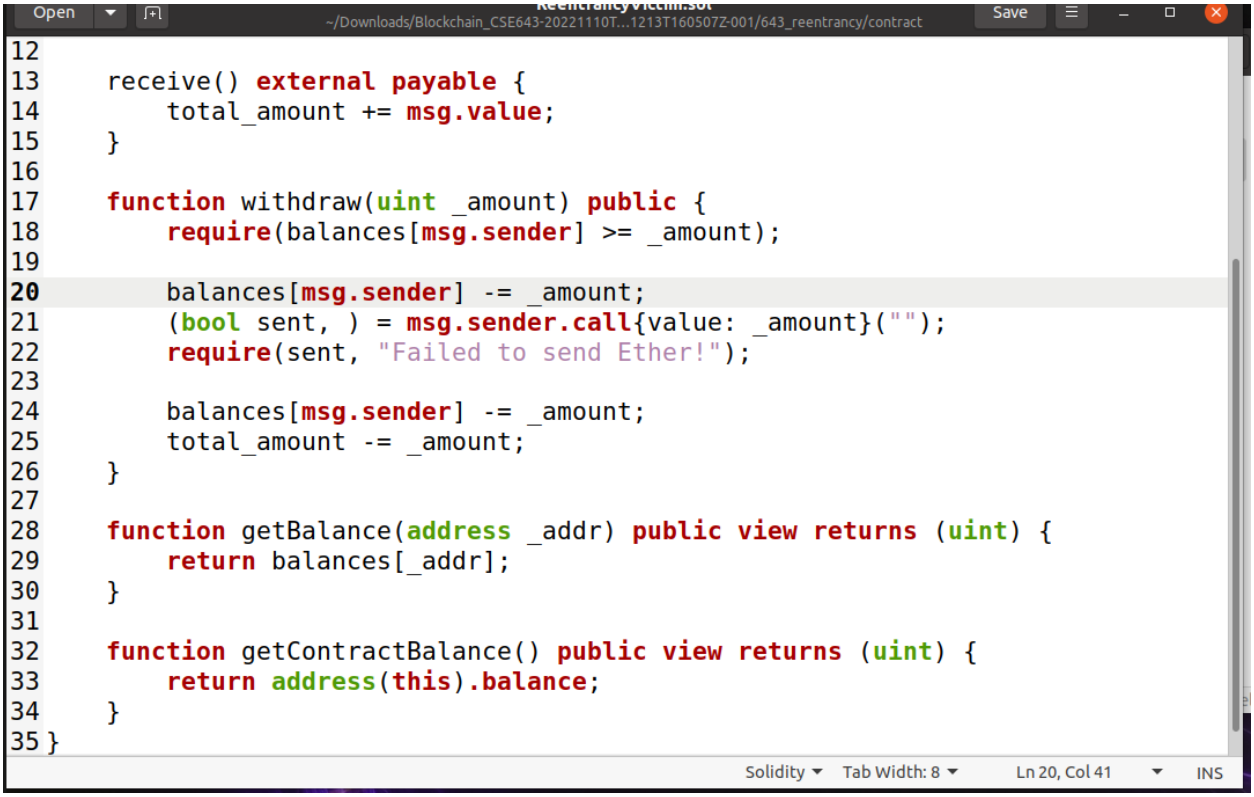
We then withdraw 5 ether from the above using code.

We generate the attack .bin and .abi files.

```
[12/14/22]seed@VM:~/.../contract$ solc-0.6.8 --overwrite --abi --bin -o . Reentr
ancyAttacker.sol
Warning: This contract has a payable fallback function, but no receive ether fun
ction. Consider adding a receive ether function.
 --> ReentrancyAttacker.sol:6:1:
  |
6 | contract ReentrancyAttacker {
  | ^ (Relevant source part starts here and spans across multiple lines).
Note: The payable fallback function is defined here.
  --> ReentrancyAttacker.sol:15:5:
   |
15 |     fallback() external payable {
   |     ^ (Relevant source part starts here and spans across multiple lines).
```

Counter Measures:

The attack works because we are not updating the balance before withdrawing the funds. Attacker raises a withdraw request and another request is generated which causes the fund to be withdrawn. The idea is if you update the balance the attack stops.
You can re-compile and rerun and the attack stops.


Open
~/Downloads/Blockchain_CSE643-20221110T...1213T160507Z-001/643_reentrancy/contract
Save
12
13 receive() external payable {
14 total_amount += msg.value;
15 }
16
17 function withdraw(uint _amount) public {
18 require(balances[msg.sender] >= _amount);
19
20 balances[msg.sender] -= _amount;
21 (bool sent,) = msg.sender.call{value: _amount}("");
22 require(sent, "Failed to send Ether!");
23
24 balances[msg.sender] -= _amount;
25 total_amount -= _amount;
26 }
27
28 function getBalance(address _addr) public view returns (uint) {
29 return balances[_addr];
30 }
31
32 function getContractBalance() public view returns (uint) {
33 return address(this).balance;
34 }
35 }
Solidity
Tab Width: 8
Ln 20, Col 41
INS