CS 5433 SP23: Homework 4

Instructor: Ari Juels
TAs: Andres Fabrega, Zhi Liu, Sishan Long
Due: 8 May 2023

In the previous homework, we got our hands dirty with solidity and Smart Contracts. In HW4, we are going to hack some Smart Contracts!

Policies

Submit your code to CMSX.

Integrity. For all assignments, you may make use of published materials, but you must acknowledge all sources, in accordance with the Cornell Code of Academic Integrity. Additionally, you must ensure that you understand the material you are submitting; you must be able to explain your solutions to the course instructor or TA if requested. You must complete this homework assignment *on your own.*

Disclaimer

In general, nearly all smart-contract technologies today are poorly (if at all) documented, are constantly changing, and suffer from broken or dead code or packages. That includes the ones for use in this assignment, which are fairly standard in the industry. **Please start the assignment early**. We do not guarantee responses from the TAs or instructors on errors in the assignment or such broken packages without at least 48 hours lead time.

1 Instructions

Your task in this assignment is to "hack" the contracts in the given code. In the last homework, you've already familiarized yourself with using Remix and Metamask, in this homework, you will get your hands dirty.

Again, you will need to get Sepolia testnet Ether from the faucets: `https://sepoliafaucet.com/`.

Interacting with the contracts: You can use remix with Metamask as "Injected Web 3" to interact with the deployed HackThisContract contracts and any contracts you write yourself when trying to hack the provided contracts. You can also use other methods to interact with the Ethereum blockchain like the hardhat, web3.js (or) web3.py frameworks.

2 Grading Policy

The points for each exploit are provided below. The full score is 13. This homework does not include partial credit.

Your hacks need to be **reproducible**. That is, if someone redeploys the contract and reproduces the steps you took (which you must specify in separate files, see below), the "hacked condition" (as described in each challenge) should hold. At the time of grading, for each contract, we will check that (1) your contract satisfies your hacked condition, and (2) reproducing your defined steps from a freshly deployed contract also satisfies the hacked condition. In other words, simply deploying a contract many times until the hacked condition is satisfied by chance will receive no credit.

Note: you should **not** change the code of the contracts directly to solve the challenges. At the time of grading, we will verify that the contracts were deployed as provided.

3 Challenges

For each challenge, you need to submit two things:

- Describe the sequence of steps you took to hack the contract in `[name]_steps.txt`, where [name] is the name of the challenge, e.g., `naive_programmer_steps.txt`. This should include the transactions you used to interact with the contract, and any code (if any) you used. Do not include the provided code in this file.
- Input the address of your deployed your contract in the file `[name]_addr.txt`, where [name] is the name of the challenge, e.g., `naive_programmer_addr.txt`

Unless otherwise specified, **deploy** all contracts with an initial value of 0 (the "Value" field above the "Deploy" button in Remix). Note that you may need to change this value during subsequent transactions of your attacks, however.

3.1 Hello_World

Description: A warm up contract to get your juices flowing. We deployed the contract `00_hello_world.sol` at `https://sepolia.etherscan.io/address/0xd509b7bded5581ce089b66b075191bb65879d8c1`, say hello to it by calling the `Hello World` function with the magic flag, where `_yourstring` should be your netid, and `_yourflag` should be the bytes32 version of your netid string.

Score & Level: 1.

3.2 Naive_Programmer

Description: A simple commit-reveal election contract. Now the election hasn't started and nothing has been set. Deploy your contract `01_naive_programmer.sol` first, and then interact with your contract, so that `currentWinner` is marked as "Bob".

Hacked condition: `currentWinner` returns "Bob".

Score & Level: 3.

3.3 Coin_Flip

Description: This contract implements a simple coin flip lottery. You job is to violate this lottery so that player 1 wins regardless of any of the flip decision parameters.

Deployment instructions: deploy with Value of 1000000000000000 Wei.

Hacked condition: player 1 is the winner of the coin flip with 100% success probability.

Score & Level: 3.

3.4 ERC20

Description: A sample ERC20 contract, with a big surprise. A naive user has deposited 0.005 Ether in this contract, and it's your job to make it yours. Your attack will have succeeded when the contract's balance falls below this initial 0.005 ETH.

Deployment instructions: deploy with Value of 5000000000000000 Wei.

Hacked condition: The balance of the contract is less than 0.005 ETH.

Score & Level: 3.

3.5 Lost_Ether

Description: A programmer has made an ERC20 with a silly vulnerability. Your job is to violate one of the invariants of an ERC20: an agent sends money to the contract and doesn't get any tokens.

Hacked condition:

Hacked condition: The balance of the contract address is not equal to the total amount of ETH.

Score & Level: 1.

3.6 Bad_ICO

Description: A greedy programmer has been impatient and created an ERC20 ICO without any thought. Your job, again, is to violate one of the invariants of an ERC20: an agent sends money to the contract and doesn't get any tokens.

Hacked condition: The balance of the contract address is not equal to the total amount of ETH.

Score & Level: 2.

4 Tips

Here are some tips based on feedback from previous years' students:

- Make sure that the compiler you use in Remix matches the header at the top of each file.
- The ERC20 challenge will require you to write and deploy your own attack smart contracts. None of the other contracts will require you to do this.
- Here is a series of YouTube videos about using Remix that may be helpful for this assignment:
  https://www.youtube.com/watch?v=xxJfQJ5bMfw&list=PLbbtODcOYIoE0D6fschNU4rqtGFRpk3ea&index=2&pbjreload=10
- **In Remix, make sure you are using the "Injected Web3" Environment when calling smart contract functions or it will not point at the Sepolia network!**

Submission Instructions

For this assignment, you will be required to submit any additional code you wrote for the challenges (Solidity, JS, Python, etc.) other than the provided code.

```
hw4-<NetID>
├── Hello_World
│   └── hello_world_steps.txt
├── Naive_Programmer
│   ├── naive_programmer_addr.txt
│   └── naive_programmer_steps.txt
└── Coin_Flip
```

```
|
|  |__ coin_flip_addr.txt
|  |__ coin_flip_steps.txt
|__ ERC20
|  |__ erc20_addr.txt
|  |__ erc20_steps.txt
|__ Lost_Ether
|  |__ lost_ether_addr.txt
|  |__ lost_ether_steps.txt
|__ Bad_ICO
   |__ bad_ico_addr.txt
   |__ bad_ico_steps.txt
```