sch-solutions-part-2

JohnnyTime's smart contract hacking course part 2 exercises WITH the solutions.

Smart Contract Hacking Exercises Guidelines

Structure

- The following repository is a hardhat project
- There are 3 main folders:
  - `./instructions` - All the instructions for the exercises
  - `./contracts` - The smart contracts of the exercises
  - `./test` - The test files to execute the exercises and complete them

Setup & Exercise Execution

Dependencies

- From the root folder run: `npm i`

RPC NODE - For Mainnet and Goerli Forks

- Sign up to Infura and get an Ethereum Mainnet URL for free.
- Copy the `.env-example` file and call it `.env` then set there your Infura URL for:
  - `MAINNET` - Ethereum Mainnet
  - `GOERLI` - Goerli Testnet

Exercise Execution

- To execute and test an exercise, run the command `npm run [exercise-name]` from the root folder
- You can always check all the exercise names in the `./package.json` file

Exercise Execution Example

In case you are working on the exercise "oracle-manipulation-2":

- Feel free to create contracts under `./contracts/oracle-manipulation-2/`
- Feel free to change `./tests/oracle-manipulation-2/tests.js` file

Execute from the root folder: `npm run oracle-manipulation-2` to check if you completed the exercise successfully.

Guidelines

Unless you've been told otherwise in a specific exercise, you may not change the EVM state with special hardhat functionalities (balances, block time, etc...), and you may not change the `before` and `after` sections in the test files.

Hardhat Basic Commands

```
npx hardhat help
npx hardhat test
REPORT_GAS=true npx hardhat test
npx hardhat node
npx hardhat run scripts/deploy.js
```