Introduction to FinTech

Homework 2

1 Announcement

- There are only 1 problem, containing 1 Python script, 1 report and 1 smart contract.
- Please notice that there is a discussion forum available on NTU Cool.
  1. Avoid sharing your answers on the forum to prevent any instances of plagiarism; You will get zero point in this assignment if engaging in such behavior.
  2. We encourage you to participate in discussions and answer questions on the forum rather than reaching out to the TA via email, as the forum allows for more immediate responses.
- If you encounter any issues with Homework 2, please find online resources and review the discussion on the forum initially. If challenges persist, contact `2b@csie.ntu.edu.tw` following the provided guidelines:
  1. Title your email `[Fintech-HW2][Summary-Of-Your-Issue]`. Please note that we will **NOT** receive emails in other formats as we have applied filters to our email system.
  2. Outline the methods you attempted previously, the resources you consulted, the steps you followed, and the results of your efforts.
  3. Note that ambiguous requests, such as attaching screenshots without proper descriptions, will not be answered.

2 Preliminaries

1. Do not fork or clone the repository directly, please use the template to create repository.
2. Rename the repository to **Homework 2** and adjust the visibility settings to **private**.
3. Clone the project to your machine and enter the homework directory: `cd hw2`
4. Run installation command: `forge install`
5. Create remapping files: `forge remappings > remappings.txt`
6. Create an `.env` folder under `hw2` folder, adding `RPC_URL` element with the **mainnet** RPC endpoint.
7. Run `forge test --mt testHack -vvv` and there will be a message in your console: `Happy Hacking!`
8. Navigate to the `Settings` tab in your GitHub repository, then select the `Secrets and Variables` section. Choose `Actions` and create a repository secret named `RPC_URL`, and the value should resemble 1

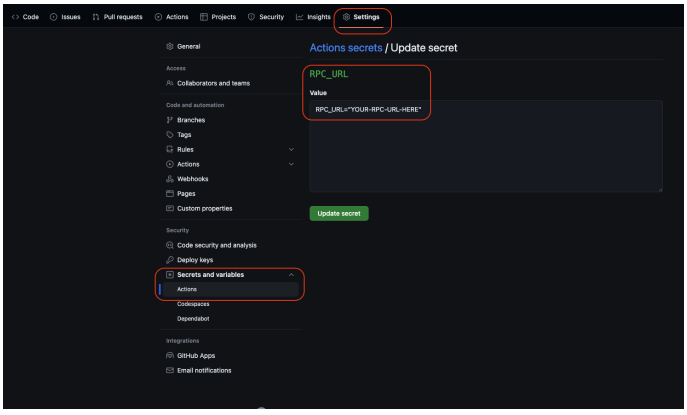

Figure 1: GitHub Secret Creation

3 Arbitrage

We have generated 5 tokens, namely tokenA, tokenB, tokenC, tokenD, and tokenE, each pair has an associated Uniswap V2 pool wherein liquidity has been provided. As an arbitrager, you possess 5 units of tokenB, and your objective is to maximize profits through the arbitrage process. You should write a simple Python script to find a profitable path and implement the smart contract. Note that the goal is not to maximize profit; any path that yields more than 20 units of tokenB passes the challenge.

3.1 Script (20 pt)

Under the root directory, there exists a Python script named `Arbitrage.py` and you need to implement an algorithm to find a profitable path. We have provided a liquidity dictionary for you, but you can still develop your own version. Note that our testing will rely solely on the provided liquidity, and you should print the returned path as follows: `path: tokenB->tokenA->tokenD->tokenB, tokenB balance=3.770765`. You will get `zero points` if you directly return a path and the balance.

3.2 Contract (30 pt)

Please check the `Arbitrage.t.sol` contract under `test/` folder. There is an `testExploit` function, please add your solutions here and do not modify any other part of the test. After finishing the contract, run `forge test --match-test testExploit -vvv` and make sure you pass the tests.

3.3 Report (50 pt)

Please complete the following tasks in the Readme file in the root repository, each problem takes 10 points.

1. Provide your profitable path, the `amountIn`, `amountOut` value for each swap, and your final reward (your tokenB balance).
2. What is slippage in AMM, and how does Uniswap V2 address this issue? Please illustrate with a function as an example.
3. Please examine the mint function in the UniswapV2Pair contract. Upon initial liquidity minting, a minimum liquidity is subtracted. What is the rationale behind this design?
4. Investigate the minting function in the UniswapV2Pair contract. When depositing tokens (not for the first time), liquidity can only be obtained using a specific formula. What is the intention behind this?
5. What is a sandwich attack, and how might it impact you when initiating a swap?

3.4 Bonus (10 pt)

Please provide the most profitable path among all possible swap paths and the corresponding Python script, along with its profit. Only the accurate answer will be accepted.

4 Note

1. The path and balance presented in the smart contract implementation, Python script output, and the report must be consistent.
2. Slight differences in precision between Solidity and Python are acceptable due to Solidity's lower precision.
3. TA will check the `Run Forge Tests` and `Run Python Script` section in CI/CD pipeline for final score.
4. If you need to install any Python package, please specify the package in `requirements.txt`