Introduction to Decentralized Finance

Homework 3

1 Announcement

- The assignment contains 5 programming problem.
- If you encounter any issues with the homework, please first visit the discussion forum on NTU COOL.
- If you need further assistance, attend the TA hour so the teaching assistant can help you with your problem.
- If you do not receive a response in the forum, or if posting your question would reveal your answer, leave a comment in your pull request (PR), and the TA will reach out to you.
- If you encounter any issues with Homework 3 that cannot be resolved using the methods above, please reach out to `2b@csie.ntu.edu.tw` following these guidelines:
  1. Title your email `[DeFi-HW3][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. Provide detailed information about your computer, including the operating system.
  3. Outline the methods you attempted previously, the resources you consulted, the steps you followed, and the results of your efforts.
  4. Note that ambiguous requests, such as attaching screenshots without proper descriptions, will not be answered.
- There is **NO** late submission allowed. After the deadline, you will automatically lose write access to the repository. Please ensure you push your code changes to GitHub before the deadline.
- You are **NOT** allowed to modify any protected files, such as `.github/**/*`, and we will provide these paths for you. You will receive zero points if any protected file is modified.

2 Preliminaries

- Create a homework repository by clicking this link.
- **Important**: Select your name and link it to your GitHub account. Failure to complete this step may result in a penalty.
- Clone the repository, enter the project directory, and install dependencies using: `cd hw && forge install`.
- Add necessary packages, run `forge install OpenZeppelin/openzeppelin-contracts --no-commit`
- **Optional**: If you encounter any issues with package linking, run `forge remappings`.

3 Problem 1: Trusted Oracle (10 pt)

An oracle acts as the bridge between the real-world environment and the closed blockchain system. One of its most essential functions is providing accurate price feeds. Many DeFi protocols, such as lending platforms, rely heavily on price data. A reliable oracle is crucial for maintaining security in these systems.There are multiple oracle providers, including `Chainlink`, `Pyth`, `TWAP Oracle`, and others. What is the difference among these oracles?

3.1 Description

- `TrustedOracle`: In the previous assignment, Charlie exploited the vulnerabilities in the UntrustedOracle and took control of it. He learned from the mistakes and now aims to build a more reliable and trusted oracle. After conducting research, he decided to use Chainlink as the price feed oracle. Please help me complete the implementation.

3.2 Judge

The tests are in `hw/test/TrustedOracle/TrustedOracle.t.sol`.

- TrustedOracle (10 pt): `cd hw && forge test --mc TrustedOracleTest --mt testExploit`

3.3 Protected Files

- `hw/test/TrustedOracle/TrustedOracle.t.sol`
- `hw/test/TrustedOracle/TrustedOracleBase.t.sol`

4 Problem 2: Rich NFT (15 pt)

Unlike traditional lending, flash loans allow you to borrow large amounts of assets within a short period. While powerful, they are a double-edged sword. Flash loans can be used for arbitrage and collateral swaps, but they can also enable wash trading, oracle manipulation, and other exploits. Several platforms provide flash loans, including Uniswap, dYdX, Aave, Iron Bank, and more. Although each platform offers unique features, they all follow the ERC-3156 flash loan standard.

4.1 Description

- `RichNFT`: How do you define wealth? Someone created an NFT that only the rich can obtain. If you hold both 10,000 `WETH` and 10,000 `USDC` at the same time, you are eligible to mint the NFT, and also unlocks all the `WETH` and `USDC` in the contract. Try getting one to prove you're rich.

4.2 Judge

Tests are in `hw/test/RichNFT/RichNFT.t.sol`.

- RichNFT (15 pt): `cd hw && forge test --mc RichNFTTest --mt testExploit`

4.3 Protected Files

- `hw/src/RichNFT.sol`
- `hw/test/RichNFT/RichNFTBase.t.sol`

5 Problem 3: MultiPair (20 pt)

Arbitrage is a strategy where a trader takes advantage of price differences for the same asset in different markets. The trader buys the asset at a lower price in one market and sells it at a higher price in another to make a profit. In DeFi and crypto, this often involves trading between exchanges to exploit these price gaps.

5.1 Description

- `MultiPair`: There are multiple Uniswap V2 pools, each with its own liquidity. You start with only 5 units of token B and want to end up with more than 22 units of token B, please find a strategy to achieve this. Hint: You might need to write a python script for finding the path, and please consider the 0.3% swap fee during calculation.

5.2 Judge

Tests are in `hw/test/MultiPair/MultiPair.t.sol`.

- MultiPair (15 pt): `cd hw && forge test --mc MultiPairTest --mt testExploit`

5.3 Protected Files

- `hw/test/MultiPair/MultiPairBase.t.sol`

6 Problem 4: Sasha (25 pt)

Exploit or arbitrage? Recently, a real-world token called Sasha Token was targeted, but the process only involved interactions with Uniswap V2 and V3 pools. An user (or attacker?) exploited the price differences between these pools to make a profit. Some argue this was an exploit, while others claim it was simply a normal arbitrage activity. What do you think about it?

6.1 Description

- `Sasha`: There are two pools, one on Uniswap and one on PancakeSwap, both containing the same tokens but with different liquidity. Starting with 3 units of DAI, how can you increase your DAI holdings? Hint: The codebase of Uniswap and Pancakeswap is almost the same.

6.2 Judge

Tests are in `hw/test/Sasha/Sasha.t.sol`.

- Sasha (15 pt): `cd hw && forge test --mc SashaTest --mt testExploit`

6.3 Protected Files

- `hw/test/Sasha/SashaBase.t.sol`

7 Problem 5: Sasha V2 (30 pt)

If the price difference between two pools is 0.001, you can gain 1 unit of the token by buying at the lower price and selling at the higher price with an amount of 1000. To increase your profit, you could use a flash loan to borrow more tokens and execute a larger arbitrage trade.

7.1 Description

- `SashaV2`: There are two pools, one on Uniswap and one on PancakeSwap, both containing the same tokens but with different liquidity. This time, you don't hold any tokens, but you can leverage the power of a flash loan. Borrow a flash loan from `Balancer` to execute an arbitrage trade.

7.2 Judge

Tests are in `hw/test/SashaV2/ShashaV2.t.sol`.

- SashaV2 (30 pt): `cd hw && forge test --mc SashaV2Test --mt testExploit`

7.3 Protected Files

- `hw/src/SashaV2/SashaV2Base.t.sol`