Introduction to Decentralized Finance

Homework 1

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 1 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-HW1][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`.
- If you don't see the `OpenZeppelin` library in the `lib` directory, run the command for installation `cd hw && forge install OpenZeppelin/openzeppelin-contracts --no-commit`.
- **Optional**: If you encounter any issues with package linking, run `forge remappings`.

3 Problem 1: LiaoToken (15 pt)

Professor Liao encourages participation in this course and plans to distribute an airdrop using an ERC-20 token (TO THE MOOOOOON!!!). Please assist him in completing the token implementation.

3.1 Description

Please complete the ERC-20 implementation in the `hw/src/LiaoToken/LiaoToken.sol` file.

3.2 Judge

All the tests are in `hw/test/LiaoToken/LiaoToken.t.sol`.

- transfer (5pt): `cd hw && forge test --mc LiaoTokenTest --mt test_check_transfer_points`
- approve (5pt): `cd hw && forge test --mc LiaoTokenTest --mt test_check_approve_points`
- transferFrom (5pt): `cd hw && forge test --mc LiaoTokenTest --mt test_check_transferFrom_points`

3.3 Protected Files

- `hw/test/LiaoToken/LiaoToken.t.sol`

4 Problem 2: NFinTech (30 pt)

During the course, Professor Liao thinks that issuing an ICO is not enough and now wants to launch his own NFT. Please help him complete the ERC-721 implementation.

4.1 Description

Please complete the ERC-721 implementation in the `hw/src/NFinTech/NFinTech.sol` file.

4.2 Judge

All tests are in `hw/test/NFinTech/NFinTech.t.sol`.

- approve (5 pt): `cd hw && forge test --mc NFinTechTest --mt test_check_approve_function`
- setApproveForAll (5 pt): `cd hw && forge test --mc NFinTechTest --mt test_check_setApproveForAll`
- transferFrom (5 pt): `cd hw && forge test --mc NFinTechTest --mt test_check_transferFrom_points`
- safeTransferFrom (5 pt): `cd hw && forge test --mc NFinTechTest --mt test_check_safeTransferFrom`
- Mix (10 pt): `cd hw && forge test --mc NFinTechTest --mt test_check_mix_operation`

4.3 Protected Files

- `hw/test/NFinTech/NFinTech.t.sol`

5 Problem 3: Loan (20 pt)

To help you better understand the underlying design of DeFi protocols, we have developed two simple flash loan services for you! However, these two protocols have critical issues. Please help us identify the issues and rescue the funds.

5.1 Description

- `BadCaller`: Initially, the protocol owner deposits 100 ether units of LoanTokens into the `BadCaller` to provide liquidity. There is a critical issue in the protocol design. Please try to extract all the `LoanToken` from the `BadCaller` contract.
- `BadLender`: The protocol has 100 ether staked to provide flash loan services. How can you withdraw all the ether without currently holding any tokens?

5.2 Judge

The tests are in `hw/test/Loan/BadCaller.t.sol` and `hw/test/Loan/BadLender.t.sol`

- `BadCaller` (10 pt): `cd hw && forge test --mc BadCallerTest --mt testExploit`
- `BadLender` (10 pt): `cd hw && forge test --mc BadLenderTest --mt testExploit`

5.3 Protected Files

- `hw/src/Loan/*`
- `hw/test/Loan/*Base.t.sol`

6 Problem 4: Delegation (20 pt)

Considering the `delegatecall` vulnerability, we have designed two challenges for you. Please try to take over the ownership of the protocol or drain the funds from it!

6.1 Description

- `Bank`: This contract uses `multicall` to execute multiple operations in a single transaction. Although you only have 1 ether, you can actually withdraw 100 ether. How can you achieve this?
- `UntrustedOracle`:The price feed is not reliant on a single source; instead, users can submit their own price feeds, and the owner will calculate the average price to determine the final price. However, did you know that you can become the owner of the contract?

6.2 Judge

The tests are in `hw/test/Delegation/Bank.t.sol` and `hw/test/Delegation/UntrustedOracle.t.sol`

- `Bank` (10 pt): `cd hw && forge test --mc BankTest --mt testExploit`
- `UntrustedOracle` (10 pt): `cd hw && forge test --mc UntrustedOracleTest --mt testExploit`

6.3 Protected Files

- `hw/src/Delegation/*`
- `hw/test/Delegation/*Base.t.sol`

7 Problem 5: Vault (15 pt)

Vault is a well-known design in yield farming and liquid staking protocols; for more information, refer to ERC4626. However, there are issues to consider when developing such protocols. Understanding these challenges will enhance your knowledge of the vault.

7.1 Description

In this challenge, there are three roles: `owner`, `user`, and `victim`. The `owner` mints `AssetTokens` and distributes nine units to the `user` and eight units to the `victim`. Currently, there is no liquidity in the `Vault`.

The `victim` plans to deposit all their `AssetTokens` into the vault contract, but you, as the attacker, discover the transaction and decide to front-run it. To make the victim deposit all eight of their `AssetTokens` but receive no share tokens in return.

7.2 Judge

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

- `Vault` (15 pt): `cd hw && forge test --mc VaultTest --mt testExploit`

7.3 Protected Files

- `hw/src/Vault/Vault.sol`
- `hw/test/Vault/VaultBase.t.sol`

8 Closing

Happy Coding!