Week 2 - Staking SC for ERC20 to receive reward

Created	@December 14, 2021 1:59 PM
Tags	

Objectives

☐ Write a more complex Smart Contract allowing to stake a ERC20 token to get rewards

☐ Learn how to write a SC interface, import and use it

☐ Learn how to define a struct

☐ Learn how to use hardhat console.log

☐ Aware some known attacks, how to defend against

☐ Write tests with typescript and run test coverage.

☐ Write deployment script and deploy on different networks.

Requirements

- Write a Staking contract:
  - Use ERC20 token A - the token from week 1 as the stake token
  - Use another ERC20 token B to rewards to stakers
  - Define Staking contract interface
  - Define `StakerInfo` with struct instead of multiple mappings
  - Have a **deposit** function to allow user stake the ERC20 token to pool.
    - Staking contract starts to mint 40 tokens B corresponding each block, from block number `START_FROM`, these tokens are used to reward to

stakers

  - Have a **getReward** function to allow a staker claim their reward token
  - Have a **withdraw** function to allow a staker withdraw their staked token
  - Inherit/implement Ownable, Pausable contracts from @openzeppelin
  - Should have all necessary events and public getters.
- Write full coverage tests for the contract.
- Write a deployment script for the contract.

Resources and Tips

- See https://solidity-by-example.org/defi/staking-rewards/
- https://docs.soliditylang.org/en/v0.8.10/
- https://hardhat.org/
  - Check out the 3rd party plugins and play around with them (Eg. contract sizer, gas reporter)
  - console.log
- https://docs.openzeppelin.com/openzeppelin/
  - Try out the contracts wizard and read / understand the different contracts being used https://docs.openzeppelin.com/contracts/4.x/wizard
-
- Check out our dao_sc repo
  - Read some of the contracts, tests (TS files, JS files are outdated) and deployment scripts (hardhat tasks)
  - Try running coverage

Extra / Bonuses

- Check out this challenge and try to hack it: https://www.damnvulnerabledefi.xyz/challenges/2.html

- Contract best practices: https://consensys.github.io/smart-contract-best-practices/