Cairo Smart Contracts and Starknet Intro Workshop

Workshop by JohnnyTime and Smart Contract Programmer

The following is a demo of an exercise in the Cairo Smart Contract Hacking Course by JohnnyTime. A Live workshop was held at the Blockchain Security Academy, where we went through the exercise and explained the solution.

YouTube video of the workshop recording: TBD.

Prerequisites

1. Install asdf
2. Using asdf, add scrab 2.6.3 as a plugin: `asdf plugin add scarb`
3. Using asdf, install scarb 2.6.3 - `asdf install scarb 2.6.3`
4. Using asdf, set as a global var version - `asdf global scarb 2.6.3`
5. Install Starknet-Foundry 0.24.0
    - Run - `curl -L https://raw.githubusercontent.com/foundry-rs/starknet-foundry/master/scripts/install.sh | sh`
    - Install version `0.23.0` | `snfoundryup -v 0.23.0`
6. Install Universal Sierra Compiler
    - Run - `curl -L https://raw.githubusercontent.com/software-mansion/universal-sierra-compiler/master/scripts/install.sh | sh`

In the following exercise, your goal is to create a simple smart contract with Storage, and Events, and the test is using Starknet-Foundry.

If you are not sure about the syntax, you can always refer to the Lecture video or to the Cheatsheet File.

Tasks

Cairo Contract Implementation

In the file `src\lib.cairo`:

1. Define an interface `IMyFirstCairoContract` with 2 functions:
    1. `set_number()` - receives a u256 `number` and returns nothing.
    2. `get_number()` - received nothing and returns a u256.
2. Create a new cairo smart contract `MyFirstCairoContract`:
    1. Define the contract storage with one `u256` variable named `number`.
    2. Define an event that is called `NumberChanged`, which will be emitted anytime the number is being changed in the storage, the event should be emitted with the old and new number
    3. Create a constructor function that receives an `initial_value` u256 and writes it to storage, don't forget to emit a `NumberChanged` event.
    4. Implement the `IMyFirstCairoContract` interface, and both `set_number()` and `get_number()` accordingly so they will set and get the number from the contract's storage, and emi$$ $$t an event in case the number is changed.

Testing Our Contract

In the file `tests\test_contract.cairo`:

1. Import all the relevant libraries:
    1. Starknet Contract Address - `starknet::ContractAddress;`
    2. Starknet Foundry - `snforge_std::{declare, ContractClassTrait, start_prank, stop_prank, CheatTarget, start_warp};`
    3. Your Cairo contract Dispatcher and Dispatcher Trait.
2. Create a new test and name it `first_cairo_contract_tests`, in the test itself:
    1. Declare the contract class.
    2. Prepare the constructor call data using `Serde`.
    3. Deploy the contract and create a Dispatcher.
    4. Check the initial value in the contract storage, and make sure it's correct (use `assert`).
    5. Update the number in storage to `1337`.
    6. Check the new value in the contract storage, and make sure it's correct (use `assert`).

Bonus: In your test file, check that the right events were emitted after the number was changed.

Checking the Exercise

To run all tests, use the command `snforge test`.

Useful links

Anatomy of a Simple contract

Contract Storage

Contract Functions

Contract Events

Running Tests snfoge