ДЗ.Межконтрактное взаимодействие

Форма: код в github

Область: Ethereum

Цель работы: получить навыки разработки и эксплуатации смарт-контракта взаимодействующего с другим смарт-контрактом

Результат: написан смарт-контракт, который вызывает функции другого смарт-контракта

Описание

В ряде бизнес-сценариев вам может потребоваться запросить информацию из другого контракта или отправить в другой смарт-контракт информацию. Для осуществления этой операции вам нужно знать следующее:

1. Вызываемая функция должна быть public или external. Это позволит вызвать его извне.
2. У вас есть информация по адресу вызываемого смарт-контракта.
3. У вас есть информация по интерфейсу функции или по сигнатуре функции. В зависимости от обладаемой информации можно использовать разные подходы к построению межконтрактного взаимодействия: call, delegatecall, interface.

Метод call

Для общения между контрактами используется ABI, который вы изучили на предыдущем занятии. Внутри языка есть специальный объект, который называется abi, в котором есть методы кодирования и декодирования данных. Это необходимо для “ручного” формирования сообщений, которые отправляются через метод call конкретного address смарт-контракта. Пример:

```
address.call{value: msg.value, gas: 5000}(abi.encodeWithSignature(...));
```

Метод delegatecall

Особенностью использования этого метода является исполнение вызываемой функции в контексте вызывающей стороны, т.е. удалённая функция захватывает своё

окружение и может изменять состояние вызываемой стороны. На основе этой особенности есть ряд атак.

```
contract A {
    uint public num;
    address public sender;
    uint public value;

    function setVars(address _contract, uint _num) public payable {
        (bool success, bytes memory data) = _contract.delegatecall(
            abi.encodeWithSignature("setVars(uint256)", _num)
        );
    }
}
```

Метод delegatecall при вызове возвращает успех или провал выполнения и результат выполнения удалённой функции.

Интерфейс

Для более простого взаимодействия смарт-контрактов используется подход основанный на интерфейсах (который под капотом работает как и метод call).
Этот подход используется, когда вы сами определяете интерфейс своих смарт-контрактов или используете общеизвестные смарт-контракты, которые предоставляют смарт-контракты типа interface. Для этого вам необходимо используя адрес смарт-контракта и его интерфейс создать экземпляр смарт-контракта и вызывать его функции. Например:

```
contract MyContract {
  address NumberInterfaceAddress = 0xa00fa8...;
  NumberInterface numberContract = NumberInterface(NumberInterfaceAddress);

  function someFunction() public {
          uint num = numberContract.getNum(msg.sender);
  }
}
```

Задание (8 баллов)

1. Разработать СмК вызывающий метод из СмК “Камень-ножницы-бумага”, обновляющее некоторое значение
2. Опубликовать код в Github

Условие

отсутствуют

Контрольные вопросы (10 баллов)

1. Почему на этапе разработки используется локальная EVM, а не рабочая?
2. Как передать eth при вызовах call и delegatecall?

Полезные ссылки

1. DelegateCall: Calling Another Contract Function in Solidity | by zeroFruit | Coinmonks
2. Contract ABI Specification — Solidity 0.8.10 documentation
3. Ethereum development environment for professionals by Nomic Labs
4. Standing the Time of Test with Truffle and Ganache
5. Delegatecall