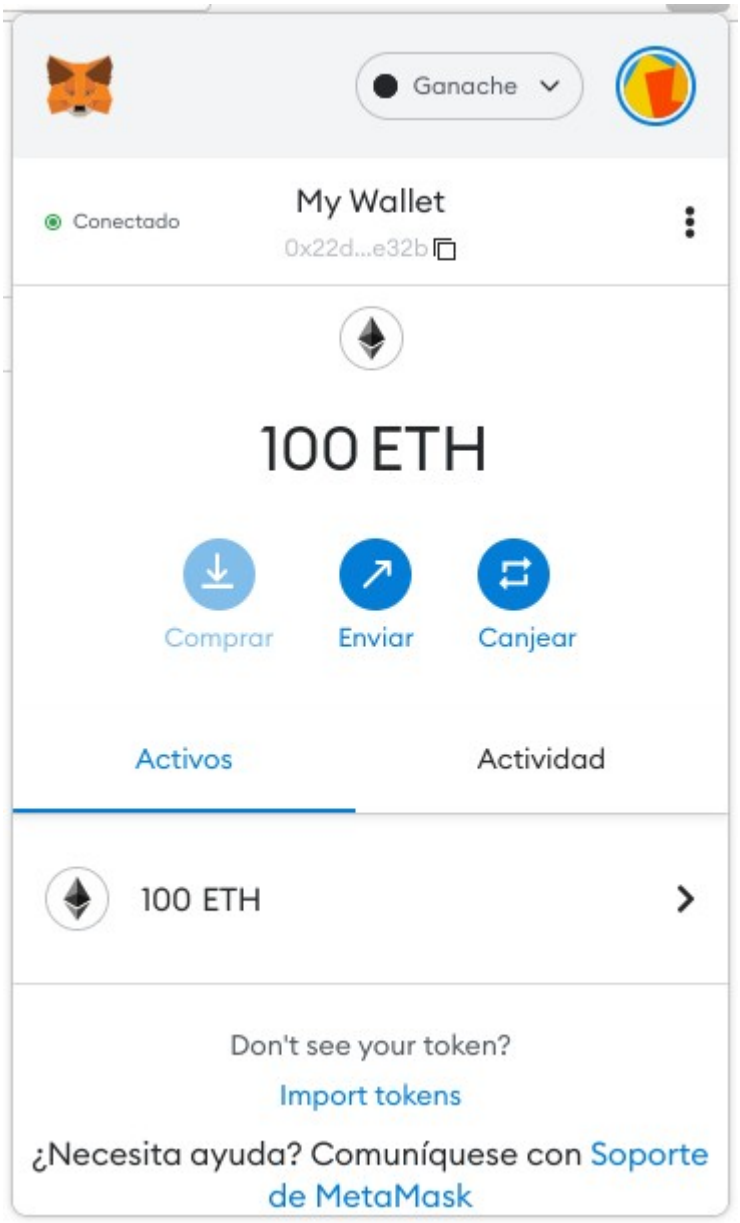
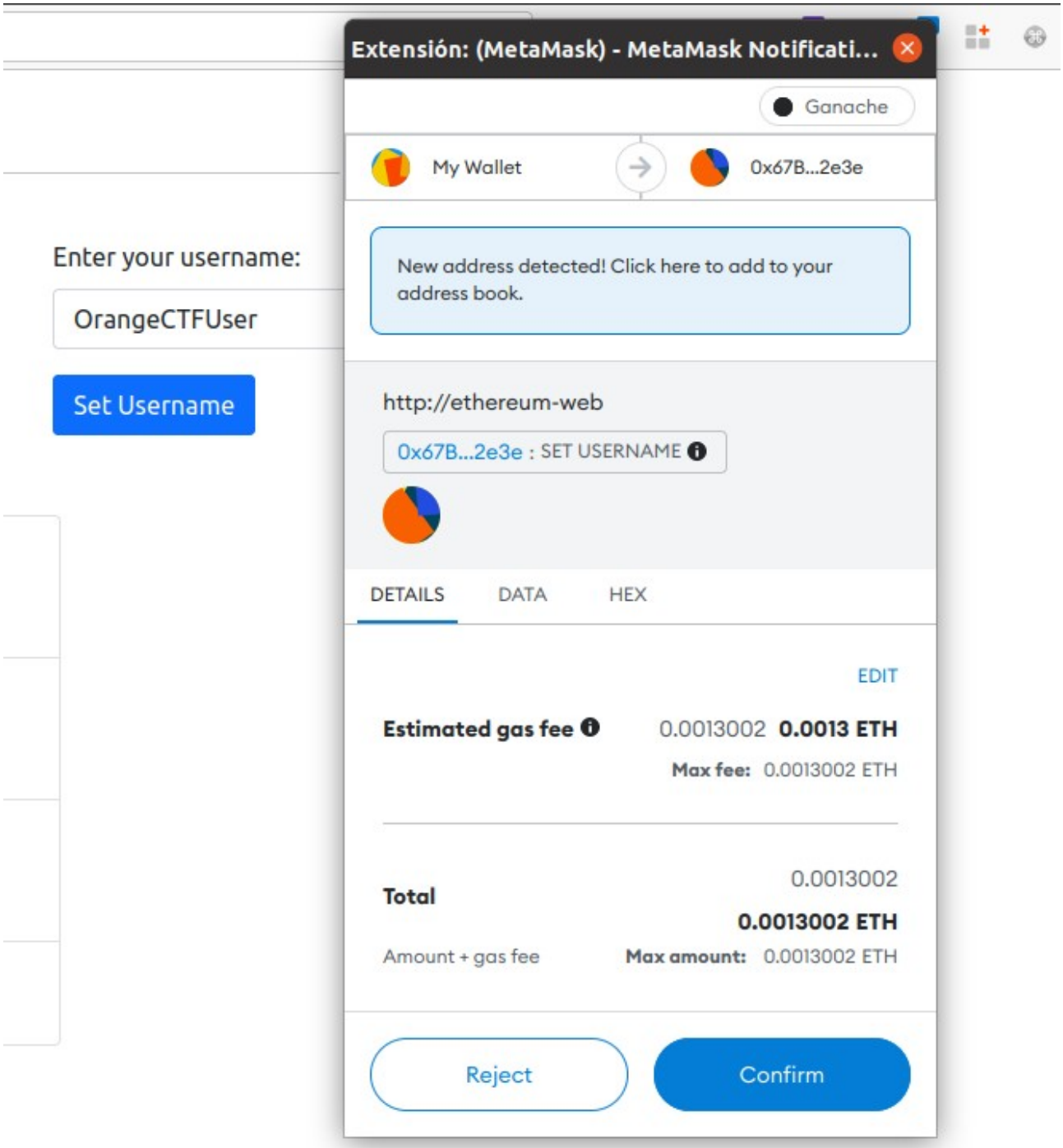
Smart Contracts Manual

First time you enter the web application you should see that you have a wallet in Metamask with 100 ETH, and you are connected to the application.

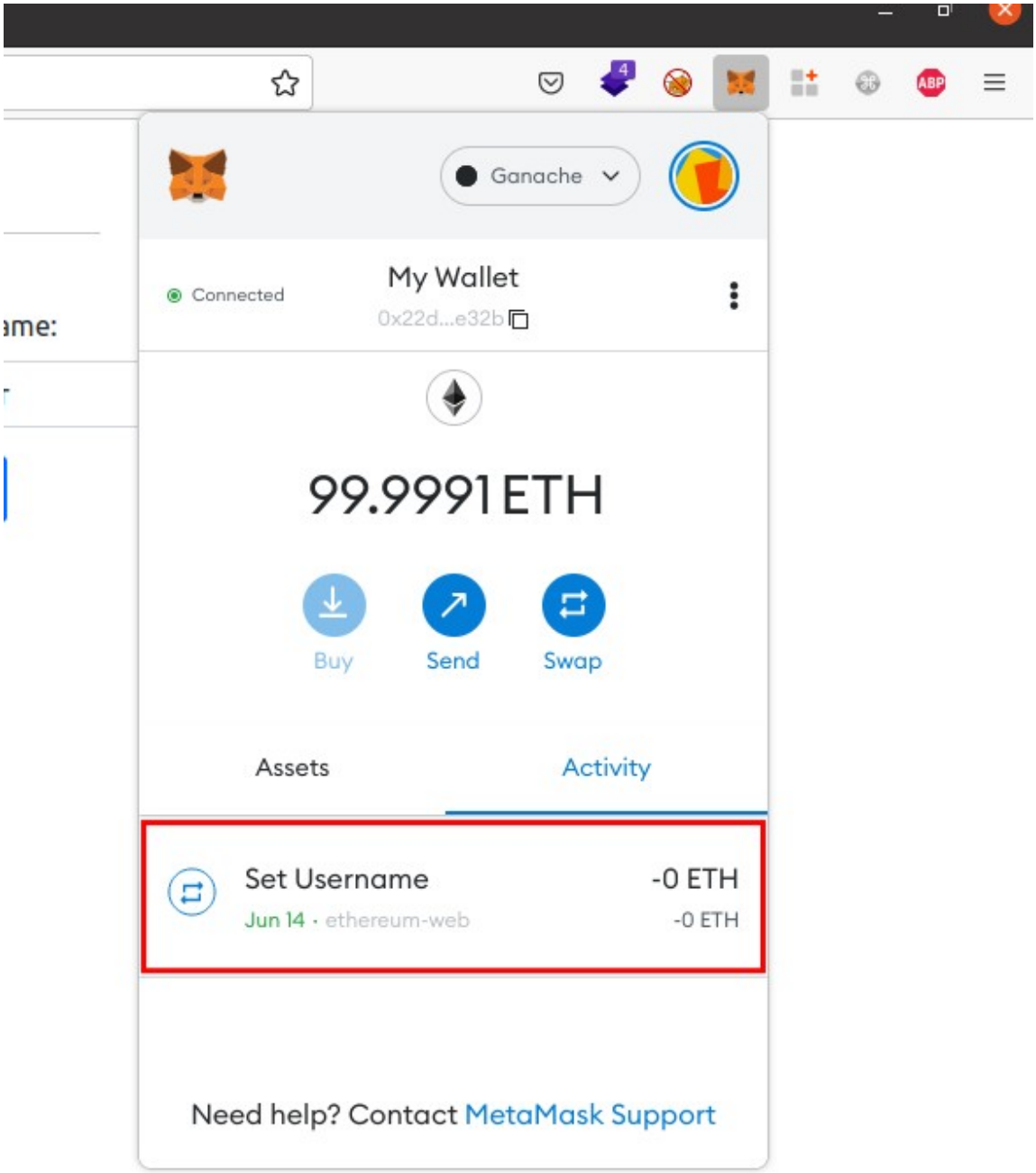


You can set your name, just to test the interaction with the blockchain, although this is not necessary.

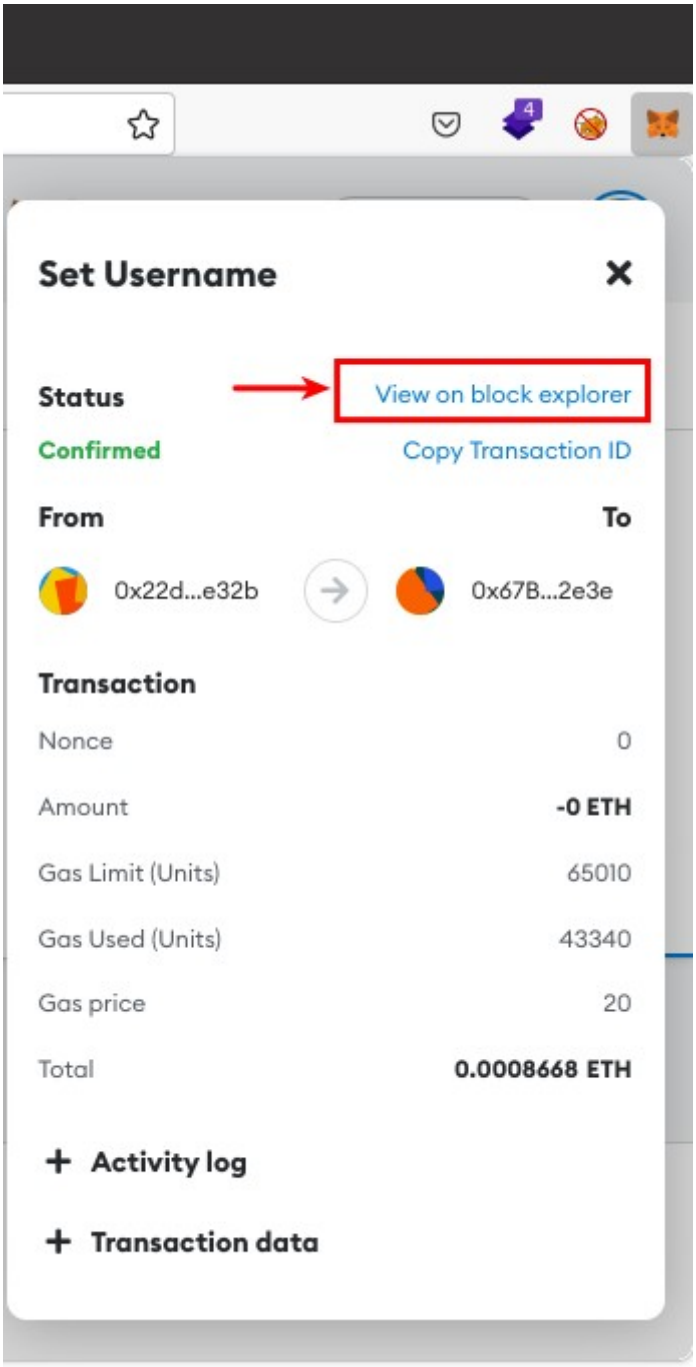


We are executing a function on a Smart Contract in the Blockchain to perform this operation. Every operation consumes a gas fee, so you would have to pay a small amount of ether.

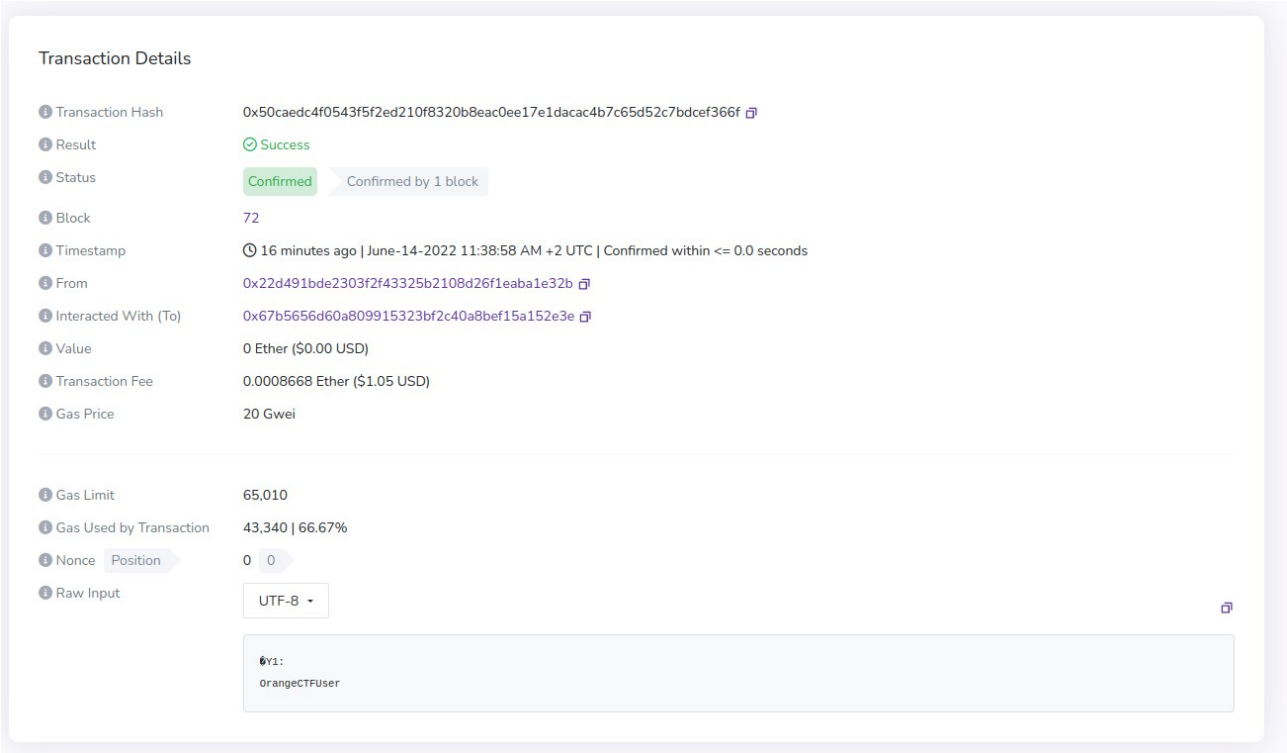
The operation has been registered in our accounting ledger:



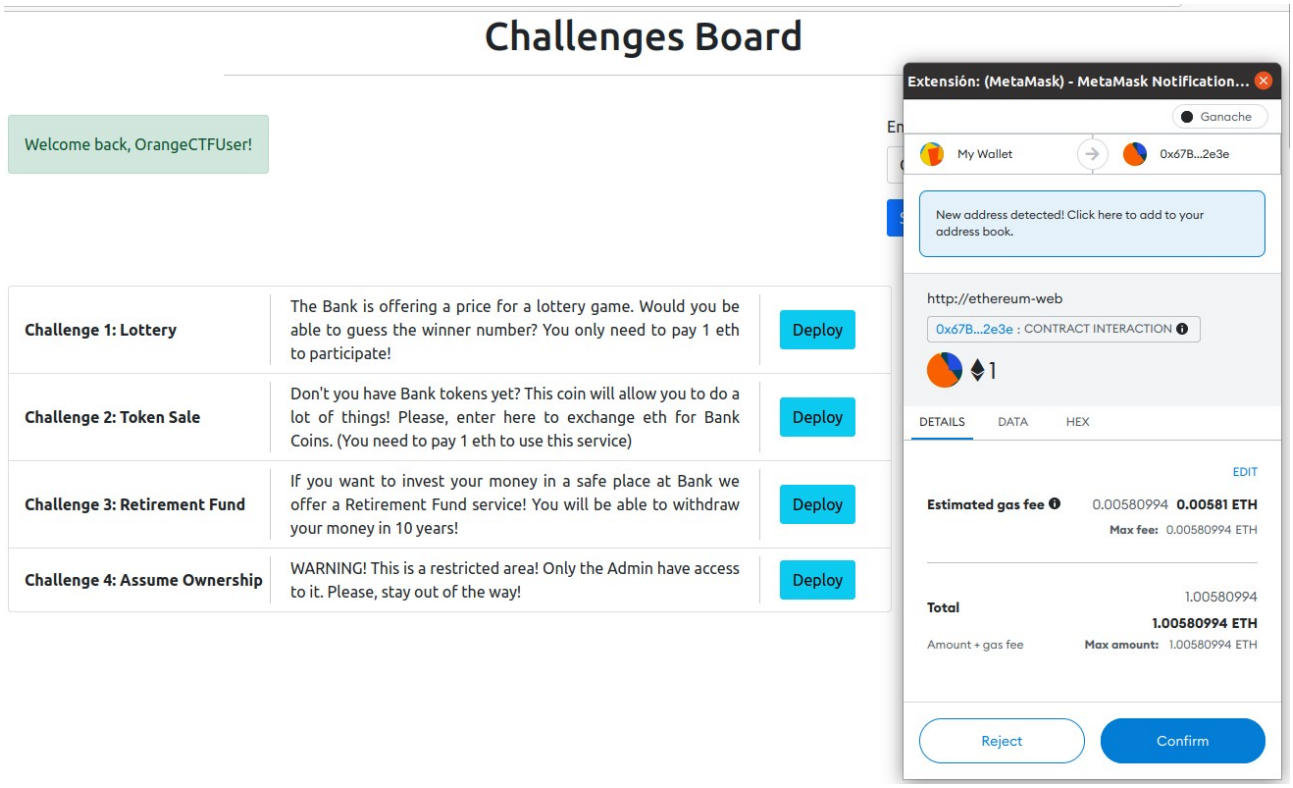
If you click on the operation block, you can see more details about it:



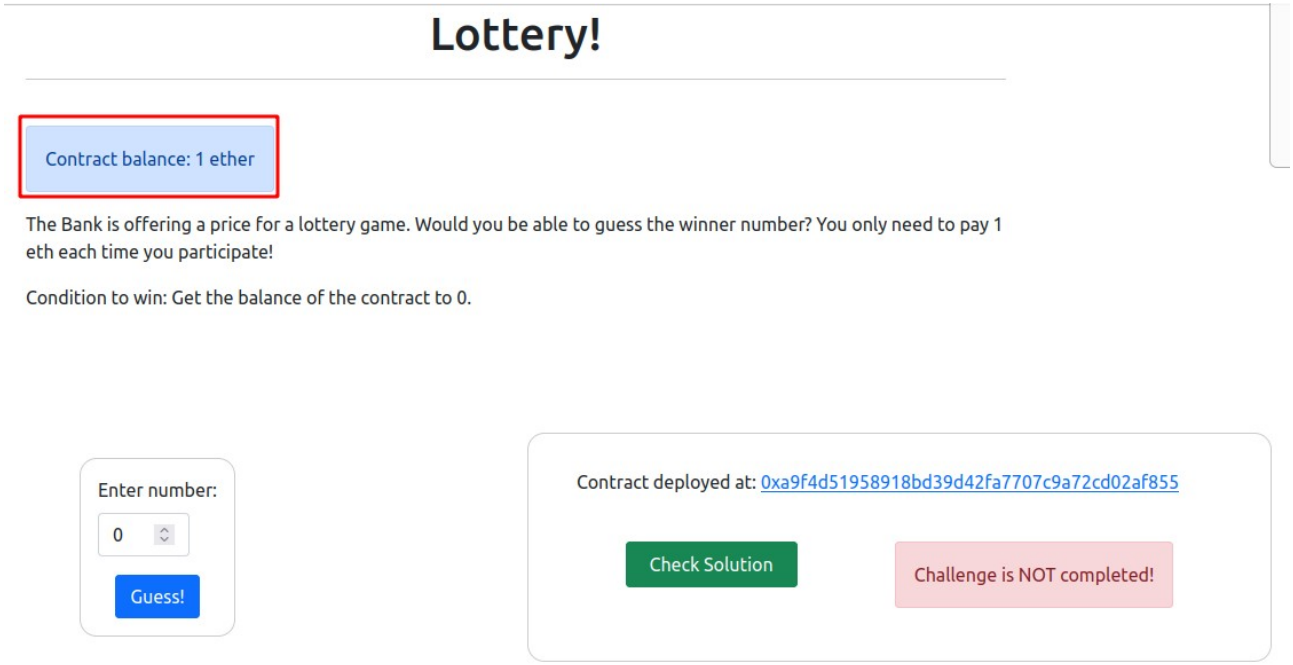
Now you can click on “View on block explorer” and you will see the transaction on the public block explorer.



Back on the Challenge Board, there is a requirement to deploy the first challenge: pay 1 ether (this amount will come back to you if you succeed in solving the challenge).



This ether is now on the contract balance:



Now we have to manage a way to retrieve back the ether!

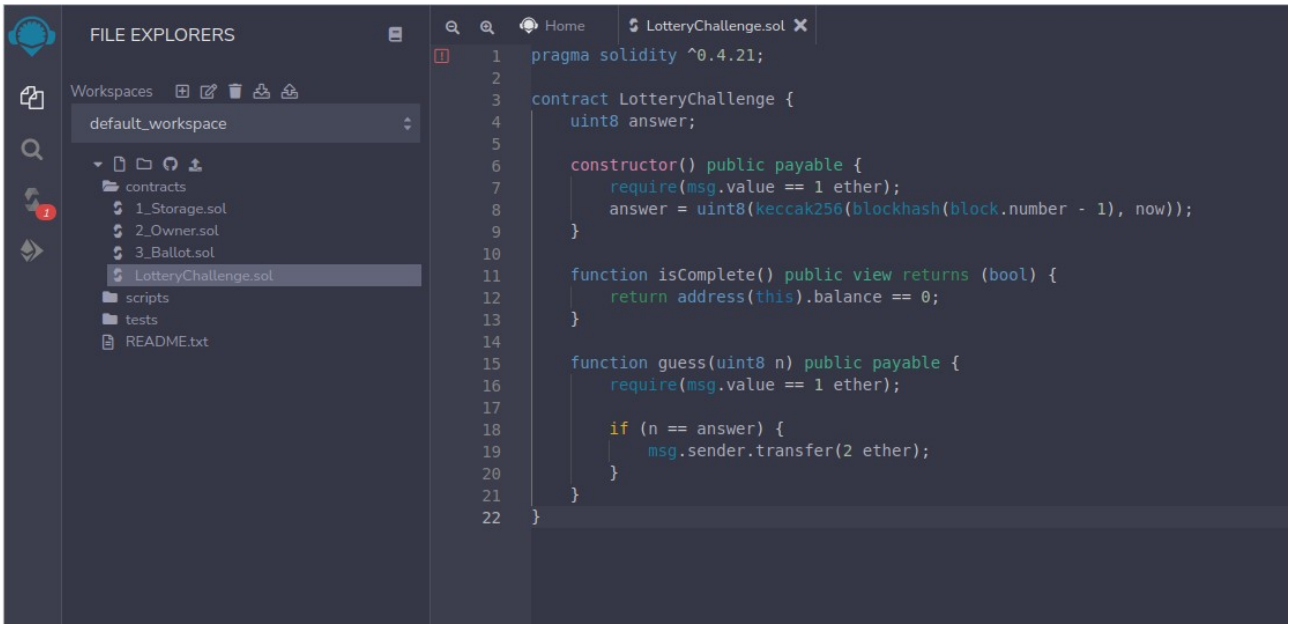
WARNING

From this part you will need access to the source code that is hidden somewhere in the website! Having the source code is crucial to interact with the Smart Contract on Remix IDE and execute our exploit on it.

Let's open Remix IDE:

https://remix.ethereum.org/

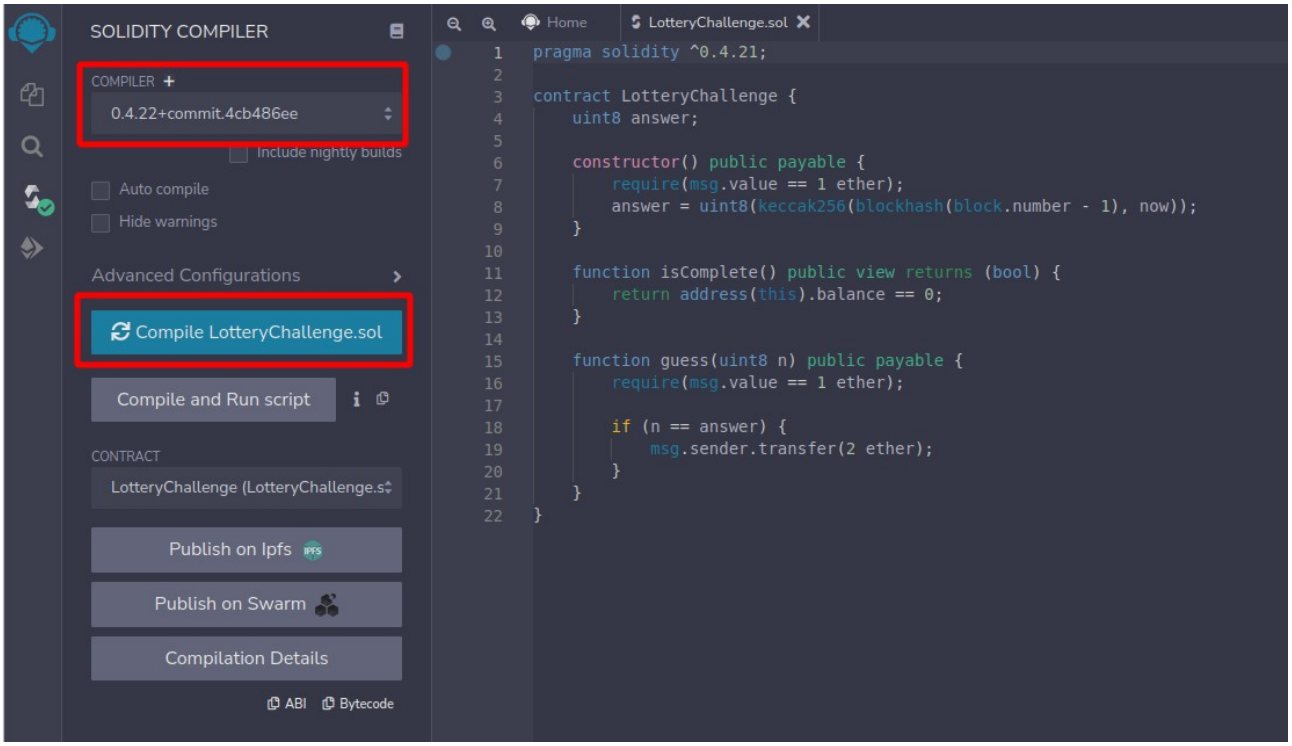
Here you have a full development IDE online for Solidity. To interact with your contract first you need to compile the code, so you have to copy the code and create a file in the contracts folder on the IDE:



Next step is compiling the Smart Contract. If you pay attention, the code is made for compiler version 0.4.1.21 and above:

```
pragma solidity ^0.4.21;
```

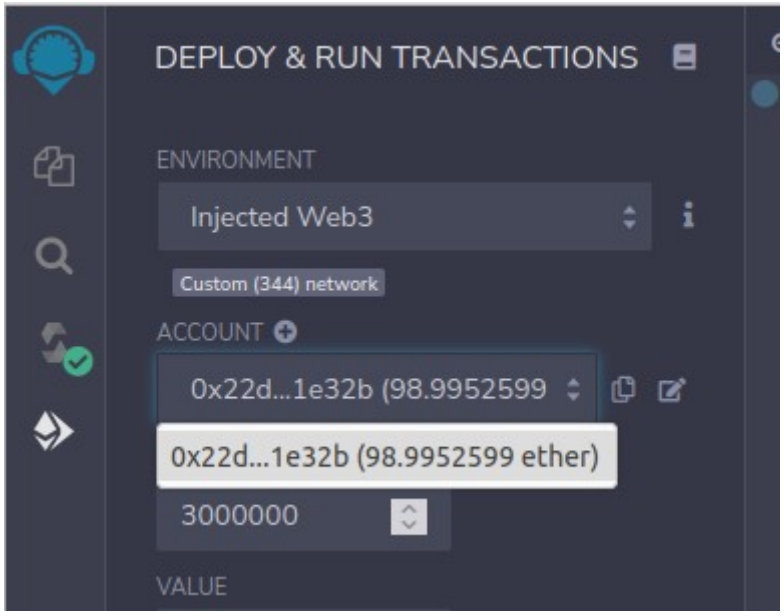
So you have to navigate to compiler section, select the proper compiler version, and click on compile. (Note: compiler 0.4.21 may not work, use 0.4.22 instead).



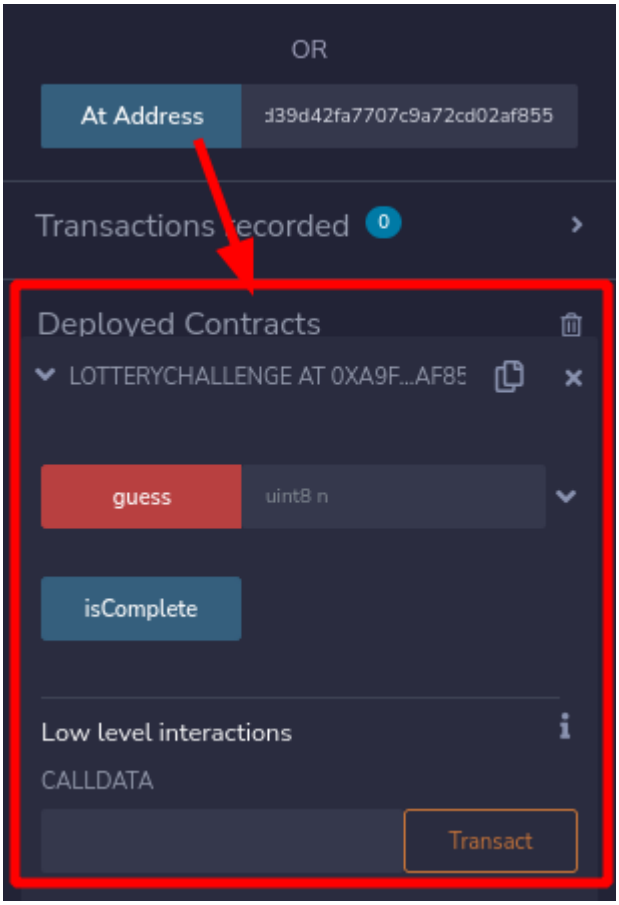
Ok, now we have the bytecode of the challenge compiled, but we have to tell Remix IDE where is it deployed in our CTF environment.

You shoud select the environment as “Injected Web3”. This tells Remix IDE that the interaction is performed through our Metamask wallet, out of the sandbox context provided by Remix IDE. Then, you will be prompted to connect your account to Remix IDE service:

If everything went ok, you should see your wallet account with your money:

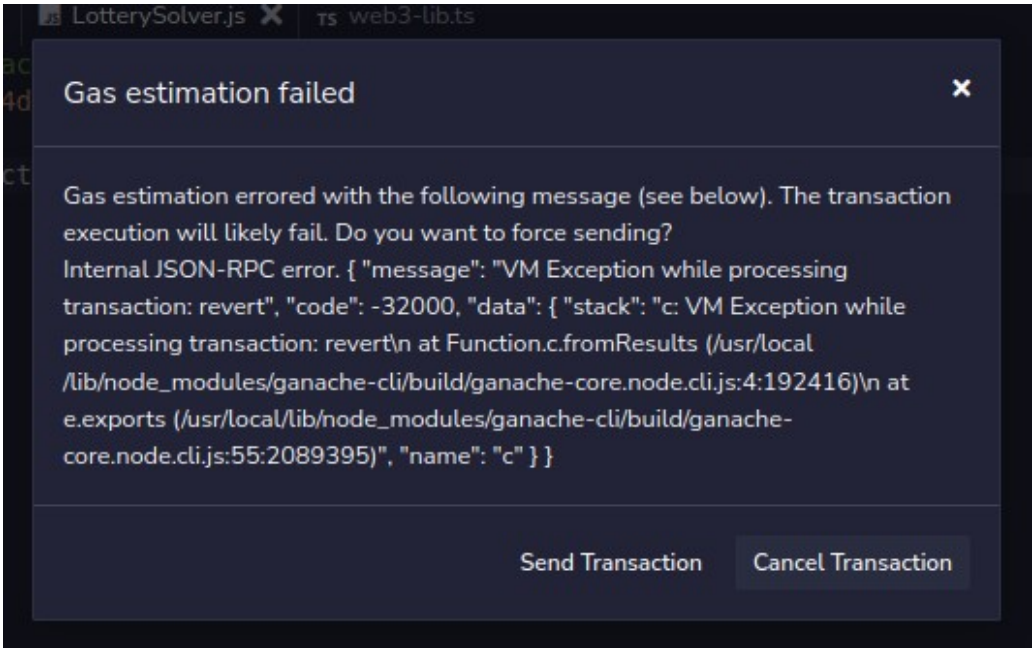


Now, we don't want to deploy a new challenge, but we want to use the one that already is deployed! So, you have to copy the address of the challenge and paste it on Remix IDE, and press the button "At Address":



Now you can start interacting with the contract from Remix IDE, and make a call to its public functions.

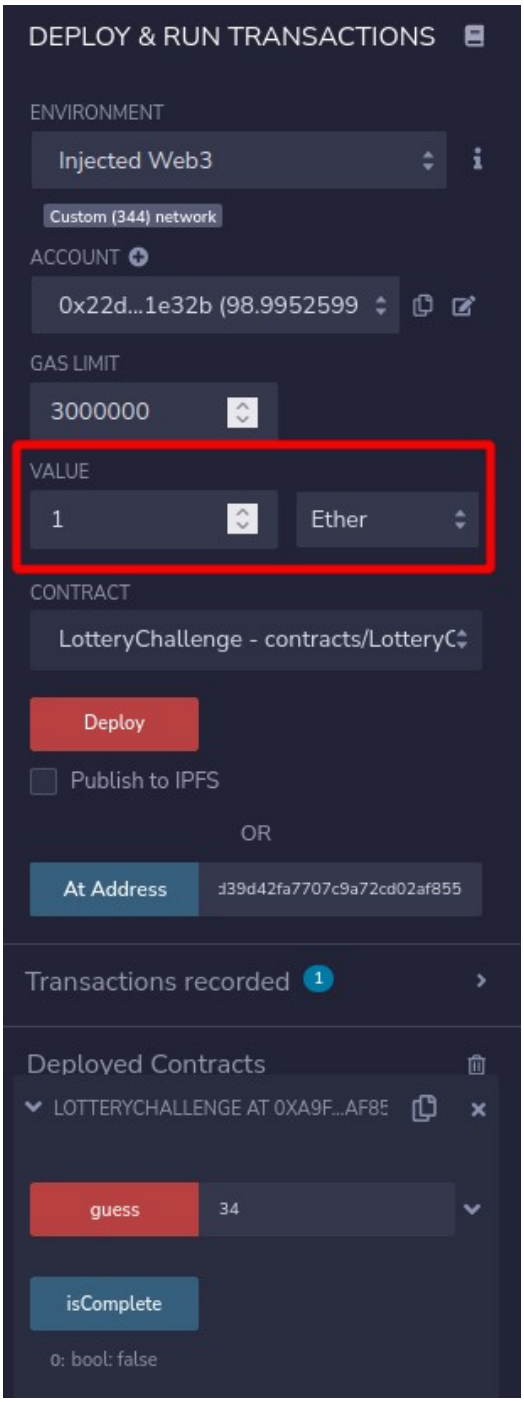
If you try to guess a number here it will give you an error. Why?



It’s because the function is expecting that you pay 1 ether as requirement:

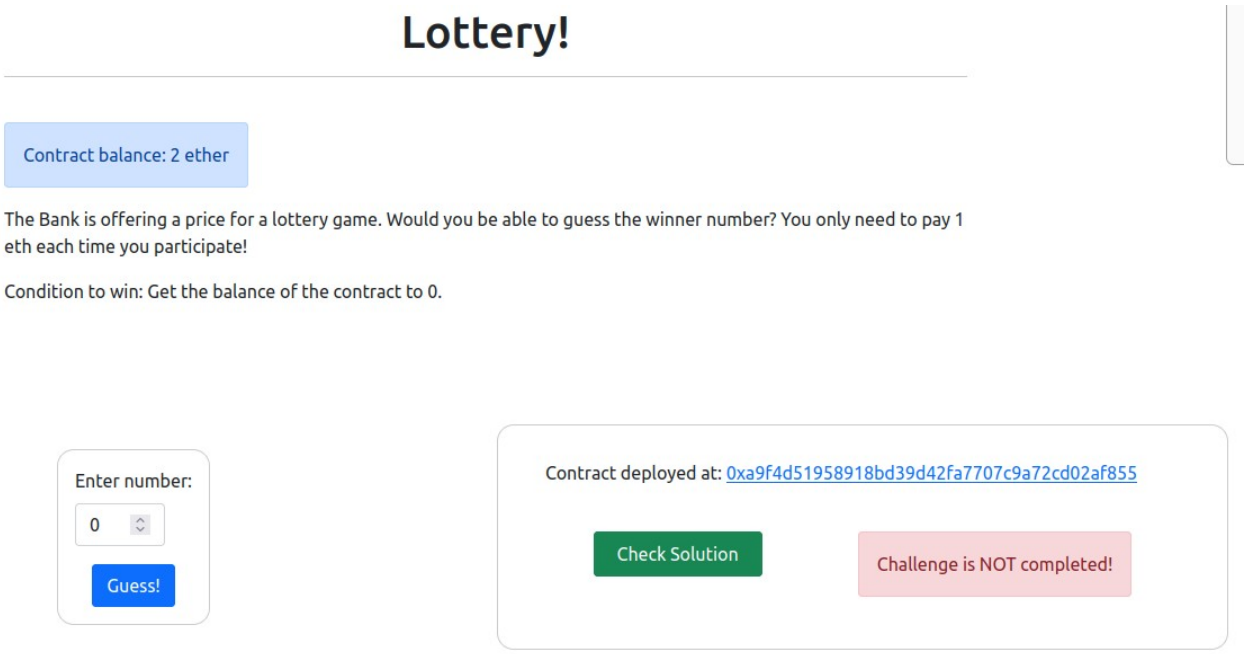
```
function guess(uint8 n) public payable {
    require(msg.value == 1 ether);

    if (n == answer) {
        msg.sender.transfer(2 ether);
    }
}
```

You can tell Remix IDE that you want to pay 10^18 wei, or 1 ether, modifying the value:



You can now enter a number and click on the “guess” button, and interact with the contract!

In the challenge we can see now that the balance has been increased 1 more ether:



You can now take advantage of the many features that Remix IDE has! Try to change the fields and the values, write a PoC using Web3.js or create another Smart Contract that interacts with the challenge. Use your imagination to solve all the challenges. Good luck!