**Smart Contract Constructor**

**Introduction:**

In this lesson, we will address a security gap present in the current fundMe contract.

**Constructor:**

Currently, anyone can call the `withdraw` function and drain all the funds from the contract. To fix this, we need to \*\*restrict\*\* the withdrawal function to the contract owner.

One solution could be to create a function, `callMeRightAway`, to assign the role of contract owner to the contract’s creator immediately after deployment. However, this requires two transactions.

A more efficient solution is to use a constructor function:

Constructor() {}

**Note:** The constructor does not use the `function` and `public` keywords.

**Assigning the Owner in the Constructor:**

The constructor function is automatically called during contract deployment, within the same transaction that deploys the contract.

We can use the constructor to set the contract’s owner immediately after deployment:

Address public owner;

Constructor() {

Owner = msg.sender;

}

Here, we initialize the state variable `owner` with the contract deployer’s address (`msg.sender`).

**Modifying the Withdraw Function:**

The next step is to update the `withdraw` function to ensure it can only be called by the owner:

Function withdraw() public {

Require(msg.sender == owner, “must be owner”);

// rest of the function here

}

Before executing any withdrawal actions, we check that `msg.sender` is the owner. If the caller is not the owner, the operation reverts with the error message “must be the owner” This access restriction ensures that only the intended account can execute the function.

**Conclusion:**

By incorporating a constructor to assign ownership and updating the withdraw function to restrict access, we have significantly improved the security of the fundMe contract. These changes ensure that only the contract owner can withdraw funds, preventing unauthorized access.