Essay – Lab 4 – Florin-George Bărcan

Once we start with the 4th chapter we get a short recap of the function modifiers that we used: *private*, which means the function is only callable by other functions belonging to the same contract; *internal*, which is similar with the only difference that the function can be called by other functions which inherit the same contract; *external*, i.e., the function can be called only outside of the contract; and *public*, where the function is callable both internally and externally. Thus we get introduced to another function modifier: *payable*. This is the bread and butter of Solidity and Ethereum: the function with this modifier can receive ether and allows for some interesting logic, like requiring a payment to the contract in order to call the function.

Here we have the function as an example for our zombie to level up:

function levelUp(uint \_zombieId) external payable {

require(msg.value == levelUpFee);

zombies[\_zombieId].level++;

}

After we send a certain amount of ether to a contract, it will stay trapped unless we write a function to withdraw the ether from the contract. Using the imported contract *Ownable* makes this easier. Using the *transfer* function, we can send any amount of ether to any Ethereum address.

After this short lesson, we refactor our solidity code for security measures and proceed implanting a zombie attack function that is payable will allow us to create new zombies based on our zombie’s DNA and the target that we attacked as well, like the CryptoKitties for example.