Cryptozombies

Chapter 4: Zombie Battle System

Contents

Payable

We're making our app pay-to-win (bleacs). We're basically setting a levelUpFee the user has to pay for leveling up his Zombie.

We need to add the payable to our function. msg.value gives us the attached ether sum.

Withdraws

We need to withdraw the real money the users spent on leveling up their digital zombies. We add the onlyOwner modifiers to our withdraw function. Owner() returns us the address of the owner which we need to cast first to uint160 and then to address. Then we can use the transfer method on the _owner object to send us the profit.

Random Numbers in Solidity

Instead of using an Oracle (external system) for RNG, we hash the timestamp, the msg.sender address and a nonce (number that gets incremented on each RNG call).

Some refactoring

Because we need to check if the sender is the zombie owner in multiple methods, we are going to create a modifier and refactor the old logic to include this modifier.

```
modifier ownerOf(uint _zombieId) {
  require(msg.sender == zombieToOwner[_zombieId]);
  _;
}
```

Using this in our old logic allows us to delete every similar line.

Attacking

We are adding two new counters inside our Zombie struct, using uint16 so they are more economical.

Also, we're incrementing them according to the RNG and the attackVictoryProbability we set