## Access Permission and Custom Modifier

## Introduction

The largest feature of blockchain is its fully public.It is always open source for its transaction even the detail like what function call during the transaction on chain,in the mean time,all users can check pending transaction , code for each contract and token.If someone leave some back door or vulnerability in their contract,that will make it exposed to criminals who try to attack and steal Ethereum.Maybe some careless programmer forget to set limitation for their function so that external personnel can directly call the owner function. This part,it is about access permission in solidity.About custom modifier,this is also pretty useful technology in solidity,maybe better to introduce with code because this vulnerability usually affect the key work “onlyOwner”.

## code demo

Because smart contract are totally open source on blockchain,to distinguish and make it easy to manage.Solidity allow program to customize modifier,which is also accept input parameter.Quite a lot developer use a common modifier “onlyOwner”

If the function is defined with onlyOwner,this means before run the code in function,it will check the msg.sender’s address.If it is the owner(may be some contract will allow multiple owner),go continue into the function otherwise revert.

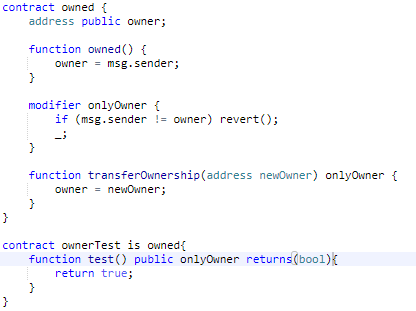


Figure 1: basic model of onlyOwner

From figure1,it is a general purpose code structure.It is always applied as a father class,once there is other contract inherit,programmer can use onlyOwner to modify function.Every transaction call test method need to pass onlyOwner check;

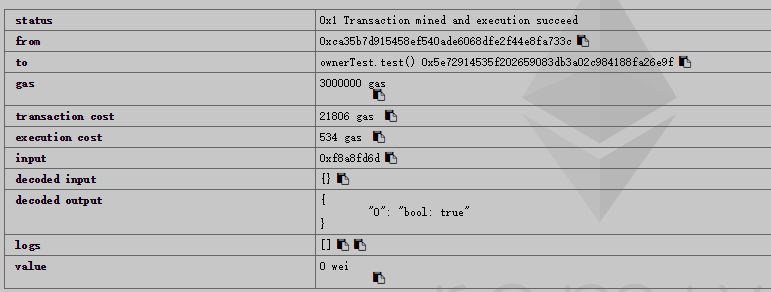


Figure 2:test with owner address

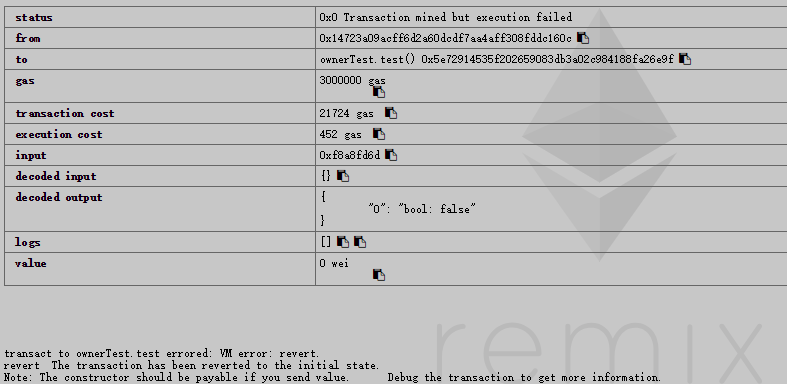


Figure 3:using non-owner address to test

As shown in figure 2 and 3,first switch the address to the contract creator,also the owner at the same time in this demo,then call the test function.Obviously,pass the modifier and return true,meaning transaction success.While the second test,using non-owner address to call,only get the false and revert.

Besides this model,there are kind of custom modifier which normalize and set rules for smart contract.While the code is fully open source,but the developer need to set some function to manage or maintain their contract,also never let stranger to abuse this part,such type of modifier become a best choice.In general,the modifier makes the function can only be accessed by owner but not unrelated persons,so how does vulnerability?

## ceoAnyone

## 3.1Introduction

Starting from the end of 2017, blockchain-based crypto-games have become popular especially with the initial success of CryptoKitties. Among crypto-games, cypto idle game is an interesting category that enables players to make money by idling for hours, then followed by a profit-making transaction (e.g., selling a Lab Rat on Ether Goo). Many of the cypto idle game owners make profit from the transaction fee. However, what if the owner address could be manipulated or completely hijacked by attackers?

## 3.2 Real World Case

This is one of contract affected by ceoAnyone:

<https://etherscan.io/address/0x5088b94cf8a1143eb228b6d3f008350ca742ddc2#code>

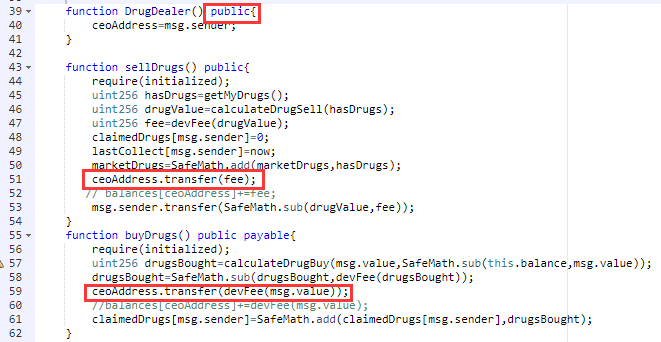
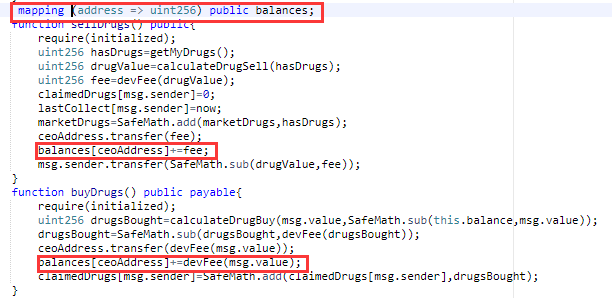


Figure 4 :A ceoAnyone-affected Crypto Idle Game Smart Contract

Figure 4 illustrates such a smart contract of a crypto idle game named Ether Cartel. Similar to [Ether Shrimp Farm](https://dapptotal.com/) that requires end users to hatch and sell shrimp in a bid to maximize production before ultimately exchanging eggs for ether, Ether Cartel takes the same concept but applies it to drug running. The game “features a high tech automated market that lets you instantly buy or sell drugs with a single transaction. The more kilos you have, the more drugs they produce (each kilo produces at a rate of 1 per day). Collect more kilos with your drugs to multiply your production.” The vulnerable code is in lines 39-41: there’s a public function named DrugDealer() which allows the caller to change the beneficiary address — ceoAddress.the ceoAddress collects the fee whenever sellDrugs() or buyDrugs() is called. To attack this contract is really easy,just call the DrugDealer function to modify the ceoAddress make it transfer value to the address you want.

## 3.3 Attack demo

To make it convenient to observer the attack,add a public mapping variable balances to record the send of value.Using the data following the first few record in the contract given:



<https://etherscan.io/txs?a=0x5088b94cf8a1143eb228b6d3f008350ca742ddc2&p=11>

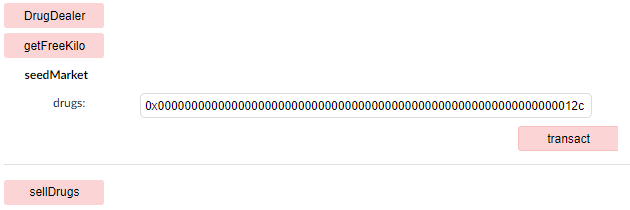


Figure5:function to set up contract

Following these step to set up contract and allow transaction:

1. create contract with address “A” 2.call DrugDealer() with address “A” 3.call seedmarket() with data in figure 5 4.call getFreeKilo() 5. call sellDrug() Then we can start normal deal:

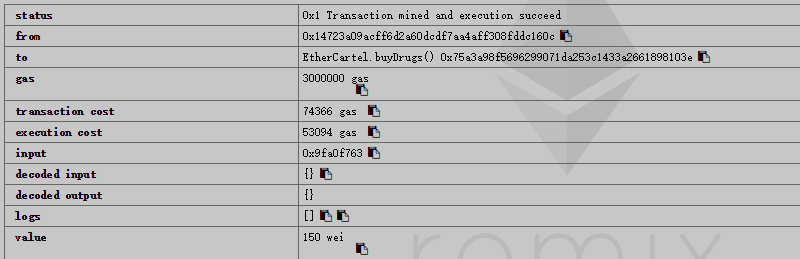


Figure 6: a success transaction

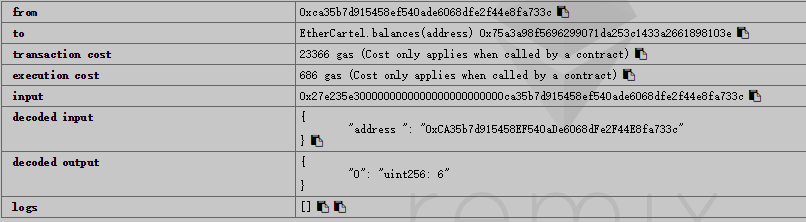
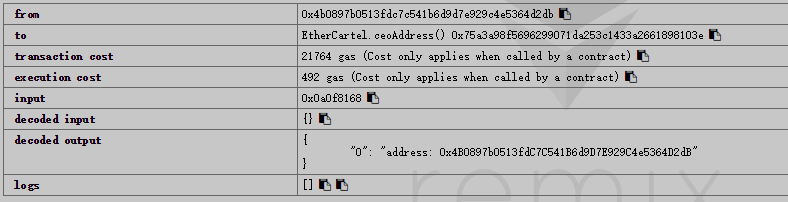


Figure 7:balance at address “A”

From figure 6 and 7,after address “B” call the buyDrug function with 150 wei value,address receive fee.Then try to attack this contract.



Firgure 8:modify ceoAddress with address “C”

After modifying the ceoAddress,do the deal from figure 6 with address B again then check the balance of address A and C

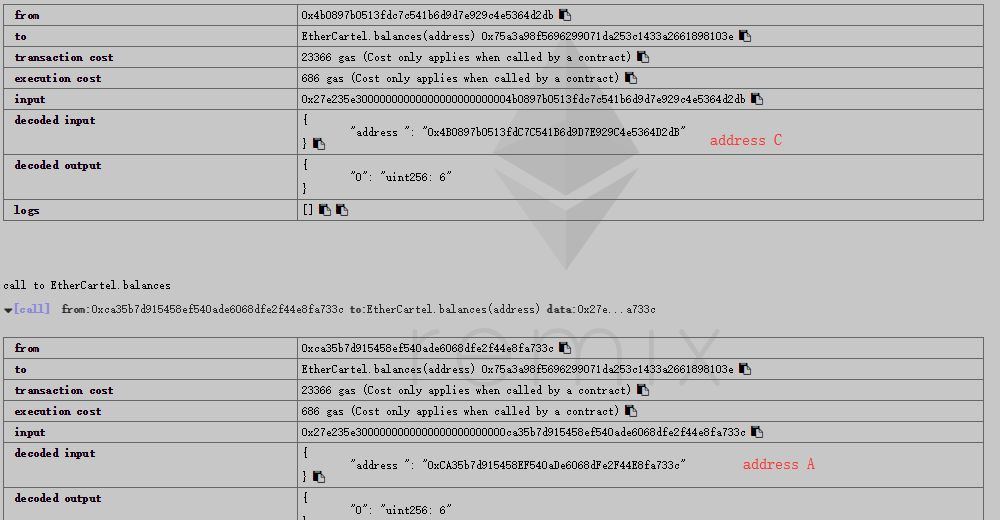


Figure 9:balance of address A and C

It is very clear to see,the second transaction transfer the fee to address C but not A.Because the function to change ceoAddress is public,everyone can easily call it to change address,causing great economic damage to the true owner.

## 3.4 Fix

It is never difficult to fix such vulnerability,just import father contract owned and modify the function DrugDealer to onlyOwner.But for these case,the structure of code seems not very comply with a standard.Here try to improve it.

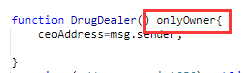
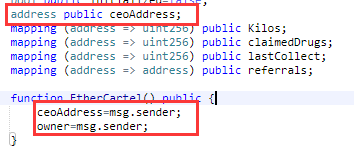


Figure 10:modification on code

In the case,the developer defines and initialize ceoAddress as default address which is not suitable.It is better to assign in the constructor.About ceoAddress and owner,maybe here is two option:1 replace ceoAddress with owner,then if owner transfer its ownership of contract,the new owner can directly obtain and get handling fee from transaction.

## 4 OwnerAnyone

## 4.1 Introduction

As we know that transactions on blockchain are run by miner.To earn more processing fee,miners always seek transaction with high gas.Even there are transactions calling same function but send by different address,miners preferentially run the one with high gas.So if some transaction initiated by contract to manage contract,but be preempted by attacker and lock the contract?In this case we will introduce the vulnerability that allow anyone to modify owner and call manager function,conflicting with the true owner’s transaction.

## 4.2 Real World Case

This is one of contract affected by ownerAnyone:

<https://etherscan.io/address/0xcdcfc0f66c522fd086a1b725ea3c0eeb9f9e8814#code>

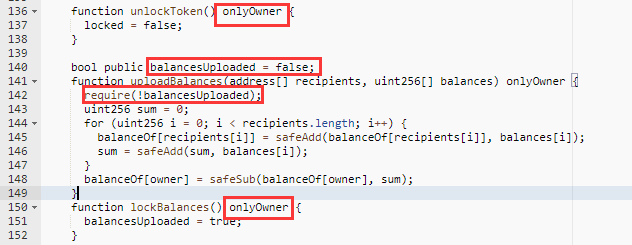


Figure 11 :part of code of AURA

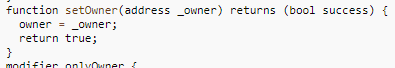


Figure 12:part of code of Owned

As shown in figure 11 and 12,in AURA contract,they set limitation with onlyOwner modifier to some function,unlockToken and lockBalance.About unlockToken,it handle the change of variable locked,which will be check in transfer and approve function ,only run while locked is in false value or called by owner.And lockBalance also control variable balancesUploaded,the function uploadBalances will only run if balancesUploaded is false.But the unlockToken function in Owned contract,developer just set as default modifier,no modifier,which will make this function can call outside like public.

Basically,the process to publish this contract on chain is like this:after create contract,call uploadBalances function to allocate balance in contract,then lock variable balancesUploaded with lockBalance function.Later release the permission to transfer by calling unlockToken function.Now we try to attack it.

## 4.3 Attack demo

After create contract,the owner is still calling uploadBalances to allocate token,maybe someone else find the transaction and contract address from chain.Before the owner lock the balance,attacker can call setOwner function to change owner then call uploadBalances to allocate balances for his address on contract.At last,before owner realize this, lock the balancesUploaded so the true can not allocate balances any more.Although such attack will not let the attackers would not be financially benefited from exploiting the vulnerability.Instead, may lead to Dos.

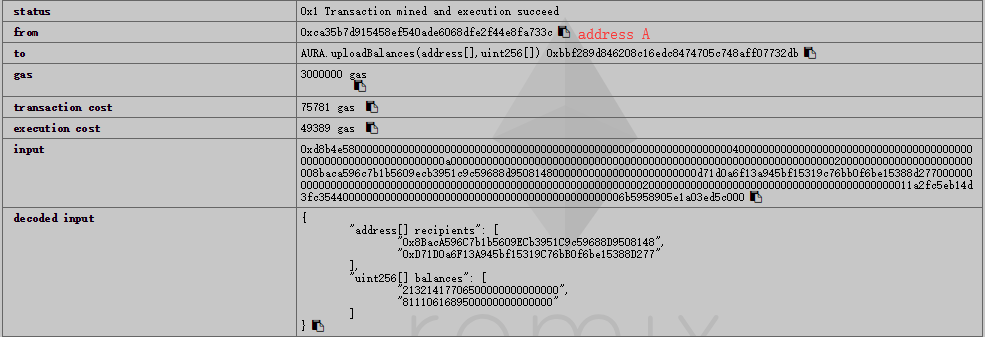


Figure 13: first time allocate balance from true owner

From figure 13,the owner start to set up balances

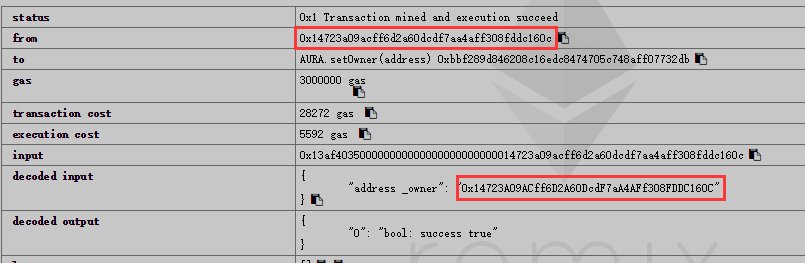


Figure 14:attacker tamper the owner address

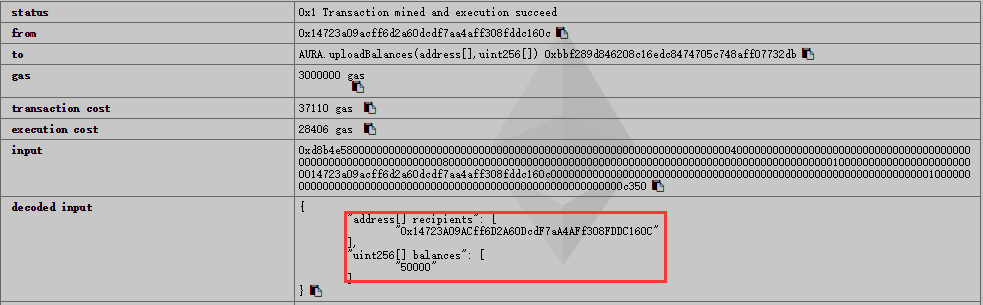


Figure 15:attacker allocate balance.

After attacker call lockBalances function,nobody can allocate balances even the true owner.

Further more,if the transaction that call uploadBalances from owner is still waiting for miners,but attacker take the lead to finish his modification with higher gas,this will the previous transaction never reach and complete.As last,that is Dos.But here,with remix-ide,it is hard to reproduce this process.

## 4.4 Fix

The same as fix ceoAnyone,and much more easier that it,in this case,just need add onlyOwner modifier in function setOwner.



Figure 16:modification on setOwner

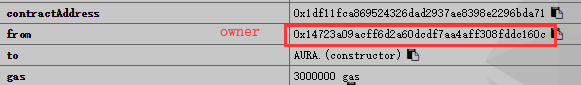


Figure 17: creation from owner

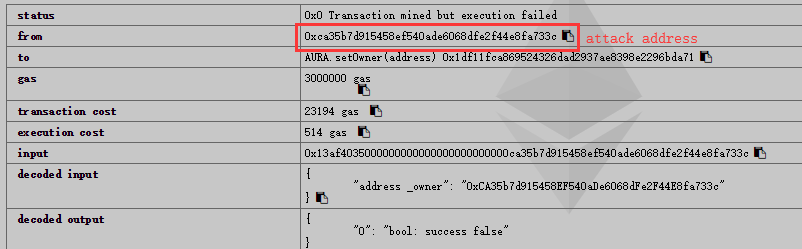


Figure 18:attacker try to tamper owner address

From figure 16 and 18,we can conclude that after modify setOwner with onlyOwner,the attacker can not pass the check to call setOwner,so that the next step to allocate balances or lock the variable is impossible.

## Summary

Writing a safe smart contract is NOT an easy job. It requires different security considerations from our traditional software development.Especially solidity allow developer to customize modifier.Different structure or detail of modifier may concern user to know about code because the same name on modifier like “onlyOwner” will have totally different implementation,which is always depended on developers.In this part,two cases show that if developer forget to check access permission before publish contract,it may allow people to tamper the owner of contract so that attacker can call function freely as they want.Sometime the time inconsis and gas of transaction may lead chaos.Besides,the transaction that with allowance will be truncated cause it suddenly has no permission to access function,resulting it keeping pending on chain.We cannot over-emphasize the importance of smart contract auditing but here from this vulnerability we can conclude some rules to obey:

1. if developers have some function that hope only be called by owner,use onlyOwner correctly by using right template.

2. before publish contract on the chain,check each function’s modifiers and visibilities.

3. if possible,try to test contract with different address by remix-ide.