

CTF新生赛之Writeup

原创

sh1kaku 于 2020-11-27 18:02:29 发布 831 收藏 4

分类专栏: [CTF](#) 文章标签: [密码学](#) [php](#) [web](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/shikaku_/article/details/110187867

版权



[CTF 专栏收录该内容](#)

2 篇文章 0 订阅

订阅专栏

CTF新生赛之Writeup

作为零基础的新生, 也是在开学后才了解了CTF, 感觉本次新生赛中颇有收获, 也是应赛制要求, 故写下这份WP, 以纪念本人的第一次CTF竞赛。

回顾和感想

回顾本次的新生赛, 难度合适, 自己从对CTF一无所知到入门再到学习相关知识, 确实有很大的进步; 从做题的过程中来看, 期间有欣喜也有挫败: 欣喜的是通过自己的努力将一道初看起来没思路的题慢慢解答出来, 在提交flag的那一刻确实很激动; 但意识到自己能做了好几道题却还只有那么点分, 或者看着在自己刚出来还有几百分的题慢慢下降直到50分, 也让我十分泄气, 感到自己的无知与弱小。但总的来说是喜悦大于气馁, 我还是在这次比赛中感受到了CTF的乐趣, 并且决定继续在这条路走下去。

关于组队

在面试那天我偶遇了自己的同学LZM (也就是本次比赛的队友), 那时候我还对CTF并不是很了解, 也不知道该如何入门。面试后我们在路上聊了会, 他说他会做一点pwn, 于是我就咨询他怎么入门, 并表明自己很喜欢数学, 于是他给了我做密码学等等建议。之后趁着双十一买了本从零到一开始看书学习, 又借着群里的资源看了CTF特训营电子版, 稍微了解了一些Web和Crypto的知识。

后来新生赛要开始了, 我的同学中没有想认真参加的 (有些参加拿学时的), 于是我问了LZM, 知道他也没有队友后我们就组成了一队, 然后我取队名叫马猴烧酒hhhhhhhhhhhhhhhhhhhh。

比赛

其实本来我觉得我队友很厉害, 但是没想到他这么厉害, 开始后三天内就把pwn杀穿了, 群里也开始对马猴烧酒有些呼声。但我才300分, 也就是做了签到题和几道简单web, 那时候心里真的很不是滋味, 明明自己没有什么作用, 但却在这个队伍里。

一段低迷后, 开始重点做Crypto的题, 对照书上的内容, 再阅读大量网上的资料, 很快就理解了RSA的概要。再观察题目的特点, 马上跑出来了前几道。但是到了后面几道, 题目就是阅读python, 这对我来说十分不友好, 因为我看不懂Python。但这时候我的队友也把PWN几乎做完了, 而他也是专攻一个方向, 没有了解过密码学, 于是我们开展团队合作: 他帮我认代码, 我告诉他题目的解法, 于是RSA1、2以及Math 1就被这么做出来了。我自己也比较开心, 因为我对团队也做出了贡献。

而最大的感触也是我们两个互补共同解答一道题, 我研究理论而我的队友负责操作, 感受到了团队的氛围。针不戳!! 遗憾的话就是没有去了解PWN和RE的相关内容, 因为队友全部做了hhh, 之后还是要广泛学习一下的!

最后再夸一波我的队友, 他超强的!! 人也好, 在我做出来超简单的题目后也会鼓励我, 55555

Misc

[签到题]关注微信公众号

这里我只做了两道签到题，不放图了吧0.0

哈哈哈送分来了

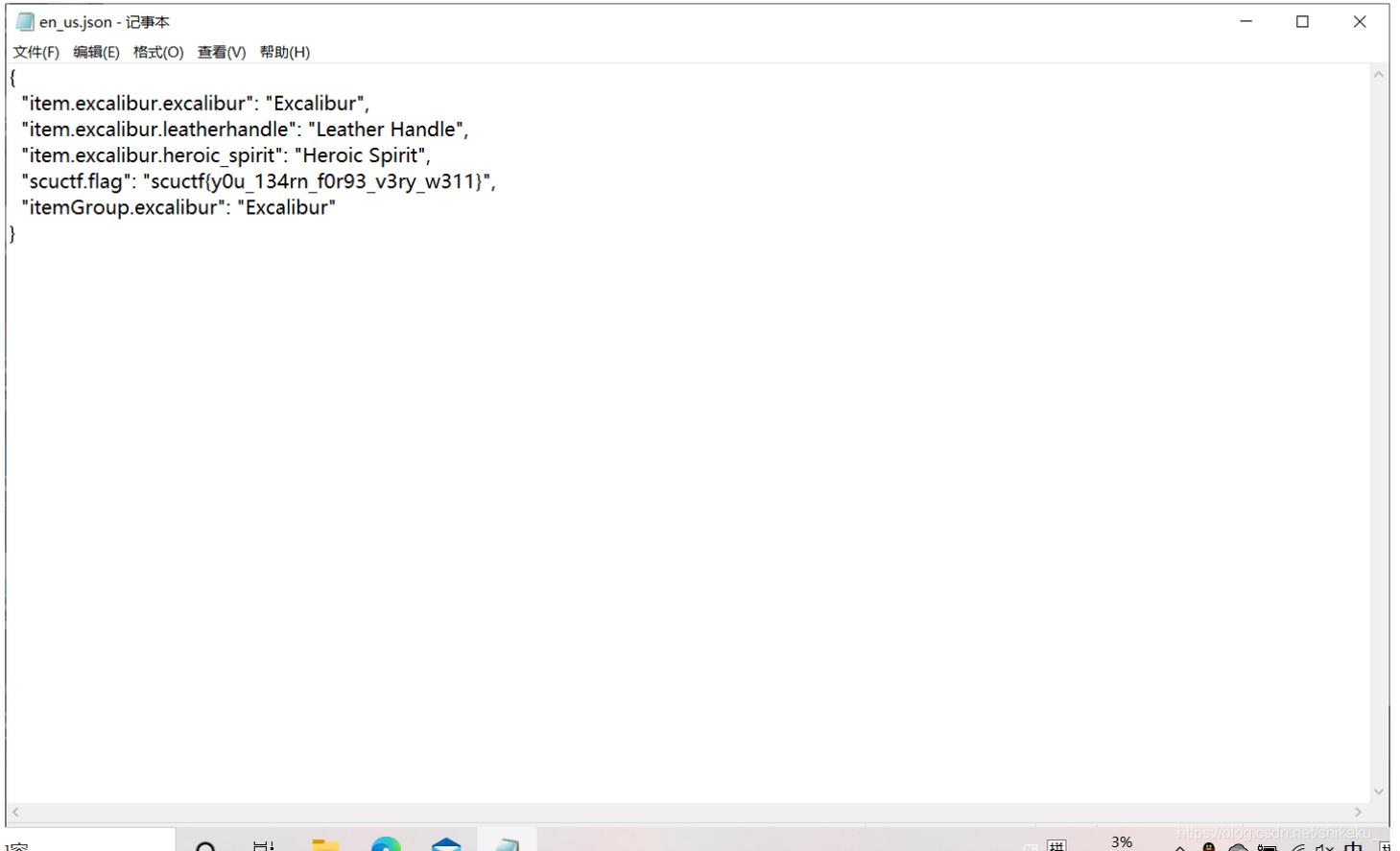
过程是关注微信公众号SCUCTF，发送SCUx401CTF即可获得flag

问卷

填写问卷，获得flag

excalibur

下载文件包后没有目的的找，突然就找到了



```
en_us.json - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
{
  "item.excalibur.excalibur": "Excalibur",
  "item.excalibur.leatherhandle": "Leather Handle",
  "item.excalibur.heroic_spirit": "Heroic Spirit",
  "scuctf.flag": "scuctf{y0u_134m_f0r93_v3ry_w311}",
  "itemGroup.excalibur": "Excalibur"
}
```

scuctf{y0u_134m_f0r93_v3ry_w311}

Web

因为是从Web入的门，所以先做的Web。这次新生赛做了六道题，都是非常简单的。但是因为看不懂PHP语言所以无法继续向前走了555，easyweb我感觉差点可以做出来，得到了hgg.com后就不会做了0.0。

真是签到题

页面一直在跳动，隐约可以看到跳动的是字符，于是想先把它停下来。登录QQ，Ctrl+Alt+A截图，便看到了flag。



情感日记

点开后的确是有很多文字，部分文字含有黄色下划线，猜测含有下划线的文字可能有线索，于是先点几次舌头，突然看到了一半的flag

你从来没说过爱我，聊天记录搜索了一下“爱”，唯一的一条是：scuctf{th1n_y0u}。

2020-11-26 成都·阴·10°C

https://blog.csdn.net/shikaku_

于是我一直点，将剩下的一半也找出来

哇哇哇，这是个啥啊，是她送给我的礼物吗咋感觉少了点什么！！scuctf{i_love，可是你到底喜欢什么啊？？？。

2020-11-26 成都·阴·10°C

https://blog.csdn.net/shikaku_

合并在一起得到flag。

easyhtml

既然和html有关，那么按F12看源码。根据页面提示“前端要写好注释”，去查看Elements，果然在body里找到了flag

学长告诉我写前端一定要写好注释呐！



(语气好温柔啊www)

我爬我爬

点开题目看到了提示“简单的robots”，于是进入网站后在后面加“/robots.txt”，进入新页面



```
User-agent: *  
Disallow: /fffl11aaaggg.html
```

看到了个disallow的后缀，代表目前还不能访问的，于是再重新加入后缀，又进入一个新页面



哇哇哇baby你太聪明了，这个小礼物快收好吧！

c2N1Y3Rme2lfZzB0X3RoZV9yMGJvdHN9

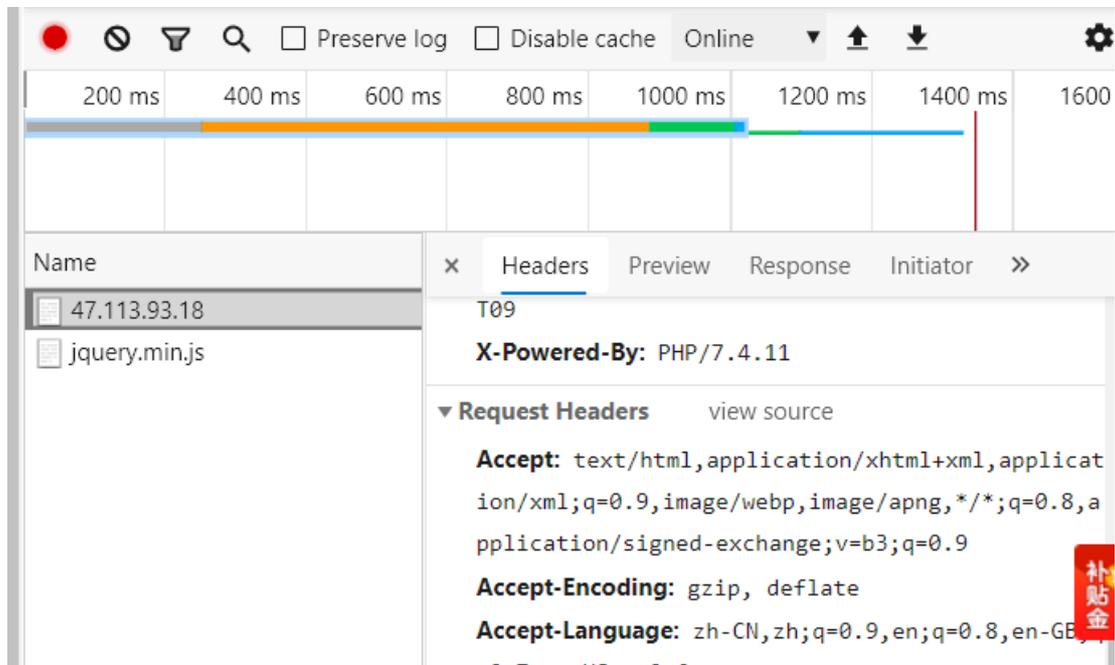
https://blog.csdn.net/shikaku_

有数字有大小写字母，base64冲冲冲，然后得到flag



easyheehee

点开发现是贪吃蛇游戏，然后看介绍游戏被破坏了，但是我还是要玩的，然后吃到第三个豆就不让吃了。我一开始想的是要修改游戏设置，让我可以吃到后续的豆，然后在Elements开始修改，一番胡乱改动之后确实可以吃后续的豆了，但是分数上不去，然后尝试修改吃一个豆的得分，然后失败了。之后去看Network，有一个很明显的cookie头



```
=0.7,en-us;q=0.6
Cache-Control: max-age=0
Connection: keep-alive
Cookie: lookme=Wm1abWJHeHNZV0ZoWjJkbkxuQm9jQT09
Host: 47.113.93.18:8005
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
```

2 requests 107 kB transferred 106

lookme都出来了，那肯定是答案了。但等号后面明显不是flag格式，猜测是加密过的，于是拿去base64翻译

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

Wm1abWJHeHNZV0ZoWjJkbkxuQm9jQT09

编码 (Encode) 解码 (Decode) ↑ 交换 (编码快捷键: Ctrl + Enter)

Base64 编码或解码的结果: 编/解码后自动全选

ZmZmbGxsYWFhZ2dnLnBocA==

https://blog.csdn.net/shikaku_

解码后面又带俩等号，那肯定又是base64加密后的了，于是再解码一编，得到一个php后缀

请输入要进行 Base64 编码或解码的字符

ZmZmbGxsYWFhZ2dnLnBocA==

编码 (Encode) 解码 (Decode) ↑ 交换 (编码快捷键: Ctrl + Enter)

Base64 编码或解码的结果: 编/解码后自动全选

[ffffllaaaggg.php](#)

解码完毕。生成固定链接

https://blog.csdn.net/shikaku_

那么把后缀加进去，就得到了flag

← → ↻ 🏠 ▲ 不安全 | 47.113.93.18:8005/ffffllaaaggg.php

flag:scuctf(hee-hee_is_really_fun)

(新生系列6)看我后面

页面提示php的备份名，直接加入/index.php没有反应，于是一个个试“.git”、“.svn”、“.swp”、“.”、“.bak”，在输入/index.php.bak后得到一个压缩文件，直接打不开，那么就进入文件夹用记事本打开

index.pnp (.)bak - 记事本

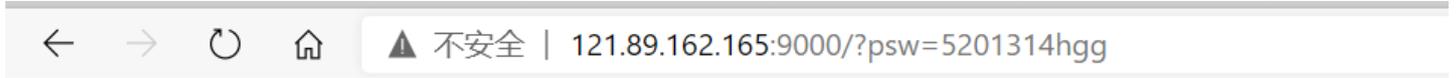
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<?php
include_once "flag.php";

if(isset($_GET['psw'])) {
    $psw = $_GET['psw'];
    if(!is_numeric($psw)) {
        exit("呜呜呜，只能是数字哦");
    }
    $psw = intval($psw);
    $str = "5201314hgg";
    if($psw == $str) {
        echo $flag;
    }
}
else {
    echo "小可爱，你知道php文件的备份名吗? ";
}
```

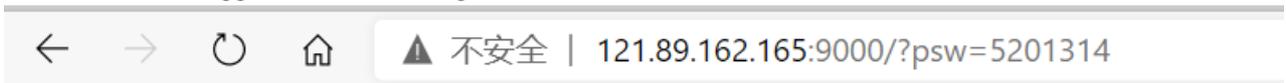
https://blog.csdn.net/shikaku_

大概看懂了是要get一个值为5201314hgg的变量psw



呜呜呜，只能是数字哦

提示只能是数字，那么把hgg去掉后就得到了flag



scuctf{it_1s_backupppp}

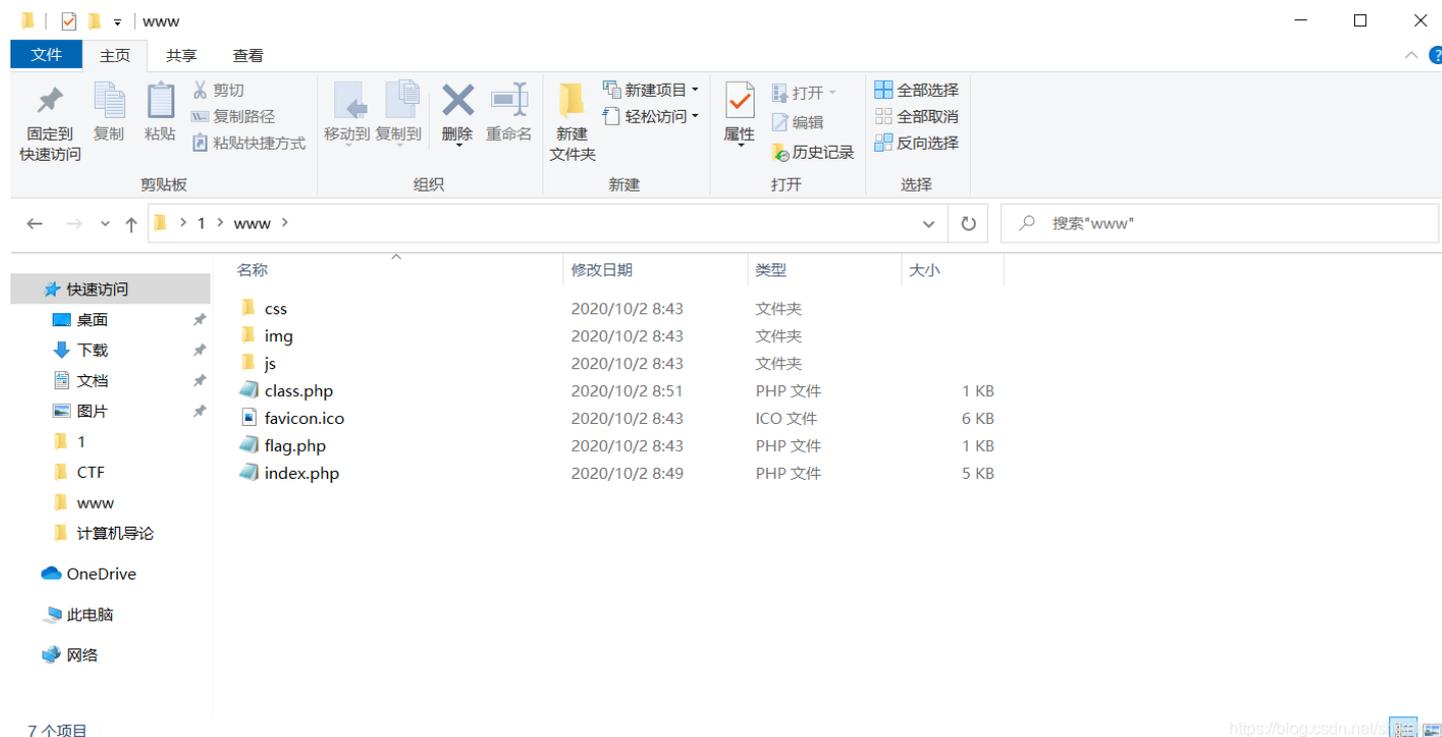
(新生系列9)渣男记录

点开网站观察动画后没什么有用的信息，然后看源码，也没看出什么（我是真的全部看了0.0），再猜是不是包含备份文件，在网站中加入后缀/index.php后没有报错，就猜测是文件包含，上网搜了一下CTF 备份文件

参考的是这篇文章

试了一下www.zip，发现真的有文件包下载，那么证明思路是对的。

将文件解压：



点开index.php观察了一下发现就是页面源码，应该没什么用；

flag.php里也没有正确的flag（我真的试了一下）；

class.php里面是一些代码，看不懂，然后和队友交流了一下。他之后就做出来了，告诉我涉及了PHP的序列化和反序列化，于是现学了一点就做出来了。

团队力量大！耶！

后记：害其实easyweb就差后面一步了，burp suite抓包后用XFF代理ip得到了hgg.com，但是进去后就不会了；以及normalphp，感觉应该不难但是读不懂代码。其实自己的web学的很浅很浅，坐等wp。

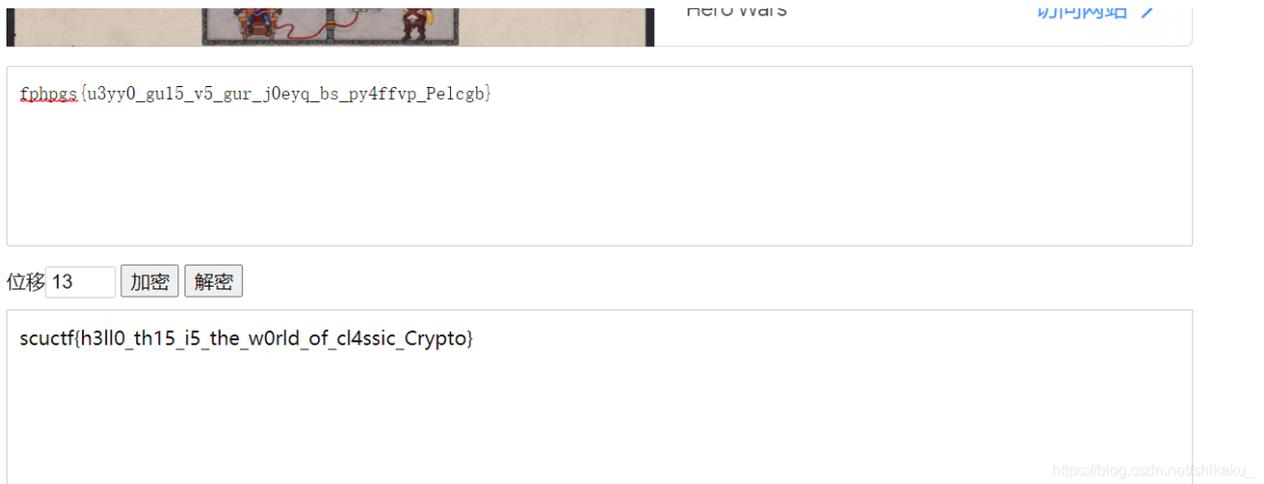
Crypto

开局放图皮一下



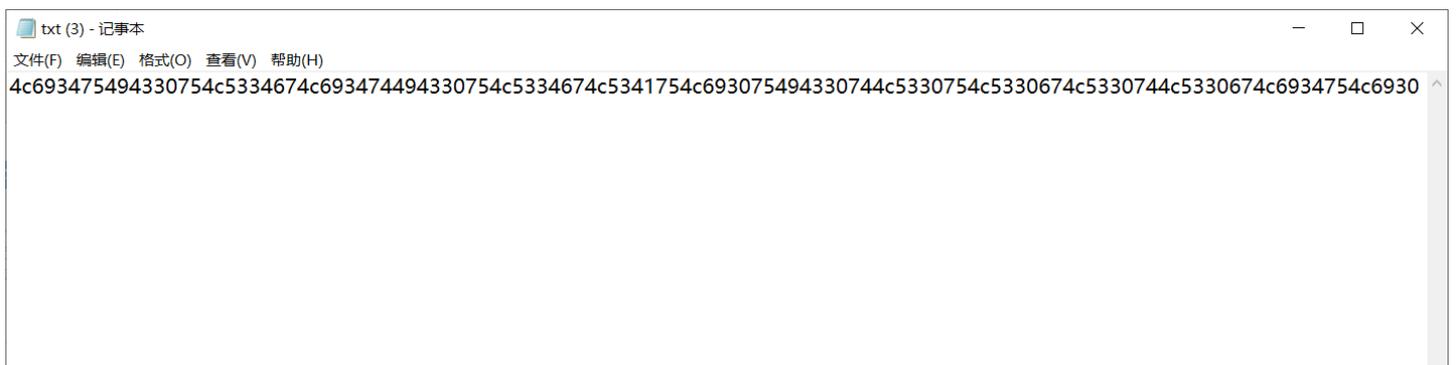
classic_1

古典密码，含有括号，凯撒密码没跑了。找个在线工具一个个试，试到13就得到了flag



classic_2

简单的多重编码，那么就是加密了多次。其实这个我一开始没做出来，因为书上的定义描述base16是数字和大写字母，但是这里是小写，所以我一开始没有想到去查base16，而base64解出来是很奇怪的东西，所以很疑惑。但后面都试过之后就没办法了，只有当作base码来解。



源文件里的密码只有数字和小写字母，根据书上的定义猜测是base16

下面就最常见的base64、base32，以及base16来进行介绍，如表19-1所示。

表19-1 三种编码字符集

编码方式	字符集
base64	a-z,A-Z,0-9,+/, 共 64 个以及补位的 '='
base32	A-Z,2-7 共 32 个以及补位的 '='
base16	0-9,A-F 共 16 个以及补位的 '='

https://blog.csdn.net/shikaku_

```
4c693475494330754c5334674c693474494330754c5334674c5341754c693075494330744c5330754c5330674c5330744c5330674c6934754c6930674c6930744c5330674c5330744c5334674c6930744c5330674c5330744c6934674c6934744c6941744c693075494334744c533074494334674c6934744c6941744c6934754c6941754c6934744c5341754c6934754c5341744c5330754c6941744c5334754c6941744c6934754c6941744c5330754c6941754c6934754c6941744c5334754c6941744c5330754c6941754c693075494334754c693475494330744c533074494334754c693475494334754c5334674c5334754c6934674c5330744c5334674c533475494334744c533074494330744c533075494330744c693475494330744c5330744c69303d
```

编码 解码 清空

```
Li4uIC0uLS4gLi4tIC0uLS4gLSAuLi0uIC0tLS0uLS0gLS0tLS0gLi4uLi0gLi0tLS0gLS0tLS4gLi0tLS0gLS0tLi4gLi4tLiAtLi0uIC4tLS0tIC4gLi4tLiAtLi4uLiAuLi4tLSAuLi4uLSAtLS0uLiAtLS4uLiAtLi4uLiAtLS0uLiAuLi4uLiAtLS4uLiAtLS0uLiAuLi0uIC4uLi4uIC0tLS0tIC4uLi4uIC4uLS4gLS4uLi4gLS0tLS4gLS4uIC4tLS0tIC0tLS0uIC0tLi4uIC0tLS0tLi0=
```

https://blog.csdn.net/shikaku_

后面有等号，又有大小写，那么是base64

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

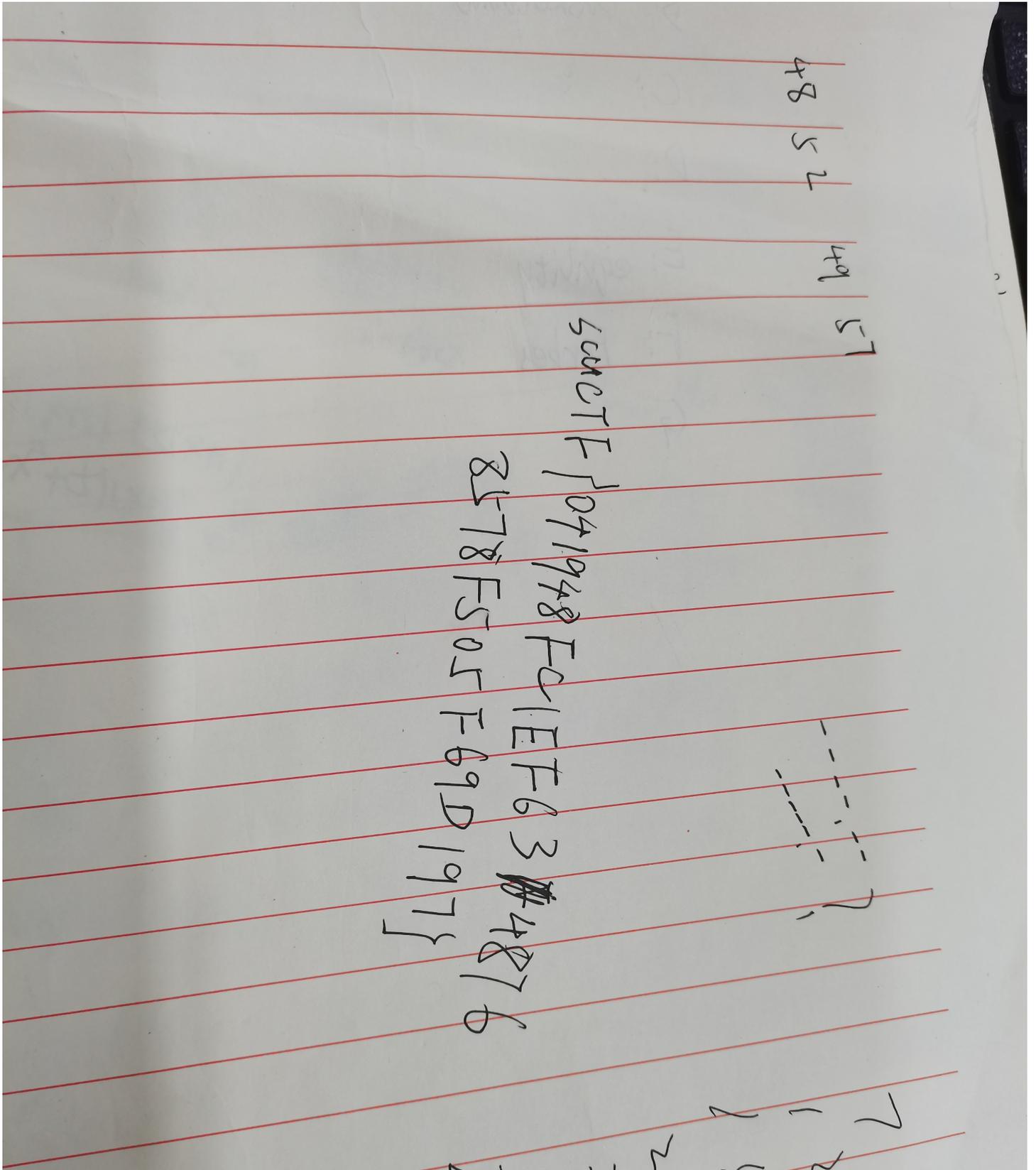
```
Li4uIC0uLS4gLi4tIC0uLS4gLSAuLi0uIC0tLS0uLS0gLS0tLS0gLi4uLi0gLi0tLS0gLS0tLS4gLi0tLS0gLS0tLi4gLi4tLiAtLi0uIC4tLS0tIC4gLi4tLiAtLi4uLiAuLi4tLSAuLi4uLSAtLS0uLiAtLS4uLiAtLi4uLiAtLS0uLiAuLi4uLiAtLS4uLiAtLS0uLiAuLi0uIC4uLi4uIC0tLS0tIC4uLi4uIC4uLS4gLS4uLi4gLS0tLS4gLS4uIC4tLS0tIC0tLS0uIC0tLi4uIC0tLS0tLi0=
```

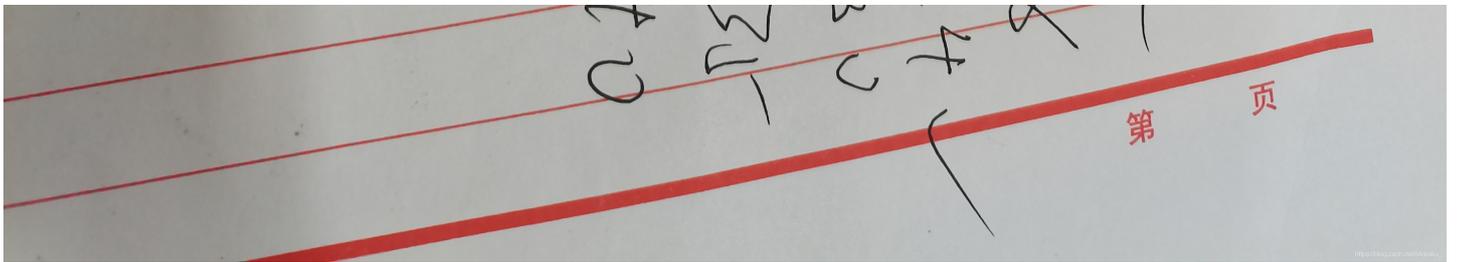
编码 (Encode) 解码 (Decode) ↑ 交换 (编码快捷键: Ctrl + Enter)

.....
.....

解出来发现是摩斯电码，当时做的时候我没找到能直接翻译的工具，于是我一个个抄下来对着摩斯表抄，抄的时候发现有两组含有6个点横，但表中没有，猜测这两个是括号，之后就解出来了。

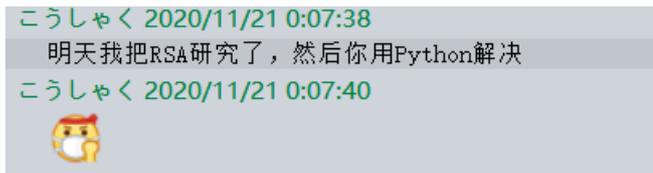
这是当时的记录：





RSA1

从RSA开始由于电脑环境以及个人水平的原因，我自己是解决不了了，于是我研究理论并且找脚本，然后让队友跑出来，这是我们做密码学之后的策略。



下载文件后看到信息

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
c = 89240837457806313116292579561992077976660301860999240662150302440032042126248255689104533304540881519976865957819
n = 17076697689025821279984148703479525857912324396375097877800474725170566885465833732966897433803722770843910606215
e = 3
```

先试着用yafu和在线网站分解n但是失败了，于是想到换个方法。因为e=3太明显了，，阅读资料后了解到这是低指数明文。如果 $m^e < n$,

所以可以知道 $c = m^3$ ，直接将c开三次方，得到的不是整数，所以 $m^e > n$ ，且没有大很多，那么存在k满足 $k * n < m^3 < (k+1)n$ ，所以可以将k爆破出来，通过关系式 $kn - c = m^3$ 来爆破明文，于是有如下脚本

```
from Crypto.Util.number import long_to_bytes, bytes_to_long, getPrime, isPrime import primefac def modinv(a,n):
return primefac.modinv(a,n) % n
n=17076697689025821279984148703479525857912324396375097877800474725170566885465833732966897433803722770843910606
2154209345260502771730300629270904051207188334736299302262170515808321795776296529107782421591087188855161497689
9585117507171481792277555517055382762767799909319596947187353003198443363190984128716735153495486042600207582210
1506835880510505034002629168205724869128357383388034971402180363910826536064357845040799329301895842061729319929
5683403344165167962678862186790420589699273314525483773243490848164411444738075659079279865450267396671572236408
48553663532280797054758912745891410981282851031085852562257 e=3
c=89240837457806313116292579561992077976660301860999240662150302440032042126248255689104533304540881519976865957
8199823419716777827460090306221618906101139272423449131313434967763664295600593363990276765767066758967765012850
672428517786210813209127040442832574380675662826137452780727264357 import gmpy2 i=0 while 1:
if(gmpy2.iroot(c+i*n, 3)[1]==1): print long_to_bytes(gmpy2.iroot(c+i*n, 3)[0]) break i=i+1
```

交给我的队友直接跑出来了得到flag:

```
scuctf{8ef98504b1acb7c8c8976e60f53d9325}
```

RSA2

已知e、n、c，那就是RSA里最基础的题了。

用网站分解n，得到三个解

Search Sequences Report results Factor tables Status Downloads

212803772175000475273337567348224776562029769705657713102085864263411671993427223373583344 Factorizel (2)

Result:		
status (2)	digits	number
FF	116 (show)	2128037721...57 <116> = 199045230832669039221046041578658179479 <39> · 319438022064098846441615805897528174851 <39> · 334688613728124045578795340681788885633 <39>

More information https://blog.csdn.net/shikaku_

然后使用如下代码：

```
from Crypto.Util.number import long_to_bytes, bytes_to_long, getPrime, isPrime
import primefac
def modinv(a,n):
    return primefac.modinv(a,n) % n
n=21280377217500047527333756734822477656202976970565771310208586426341167199342722337358334403397116963913950346969157
e=0x10001
c=7162732898109470668490761172640544970587920562229245172318483665877098759808623298921271357899945260719802967519239
p= 199045230832669039221046041578658179479
q= 319438022064098846441615805897528174851
r=334688613728124045578795340681788885633
d=modinv(e, (p-1)*(q-1)* (r-1) )
m=pow(c,d,n)
print long_to_bytes(m)
```

将代码交给队友跑出来结果如下

这个r放哪里

こうしゃく 2020/11/23 14:53:04

```
#calc
d=modinv(e,(p-1)*(q-1))
```

这里面再加个r-1

04-李钟鸣 2020/11/23 14:54:10

scuctf{e063d03aff353073c24617bd4b483f90}你提交试试

こうしゃく 2020/11/23 14:54:18

!

https://blog.csdn.net/shikaku_

Math1

04-李钟鸣 2020/11/23 21:05:52
我可以给你说一下代码的意思：找出一个正整数，它除
prime_list中的数，对应余数为c_list中的数
こうしゃく 2020/11/23 21:06:07
好
04-李钟鸣 2020/11/23 21:08:02
prime_list中的数均为256位的素数，c_list中的数为
prime_list对应数的随机非0余数
04-李钟鸣 2020/11/23 21:10:03
比如说prime_list = [7, 3, 5], c_list = [1, 2, 3], 那result =
8
こうしゃく 2020/11/23 21:12:14
(试图理解) https://blog.csdn.net/shikaku_

这一题先由队友给我翻译代码的意思

然后我想起来看到过和这个很像的一个定理叫中国剩余定理，看了一些后发现和这一题对应，于是告诉队友是这么个解法

こうしゃく 2020/11/23 21:32:40
你可以去了解一下
こうしゃく 2020/11/23 21:32:51
中国剩余定理
04-李钟鸣 2020/11/23 21:35:37
好
こうしゃく 2020/11/23 21:36:18
和这个问题很像
こうしゃく 2020/11/23 21:43:30

https://blog.csdn.net/qq_35312171/article/details

之后靠着队友强大的编程能力解了出来