buuctf - re 刷题记录 1-18

原创

yyyzzzllll 于 2021-10-29 18:52:06 发布 76 收藏

分类专栏： CTF系列问题 # re 文章标签： re

本文链接：https://blog.csdn.net/yzl_007/article/details/121040744


CTF系列问题 同时被 2 个专栏收录

36 篇文章 1 订阅

订阅专栏


re

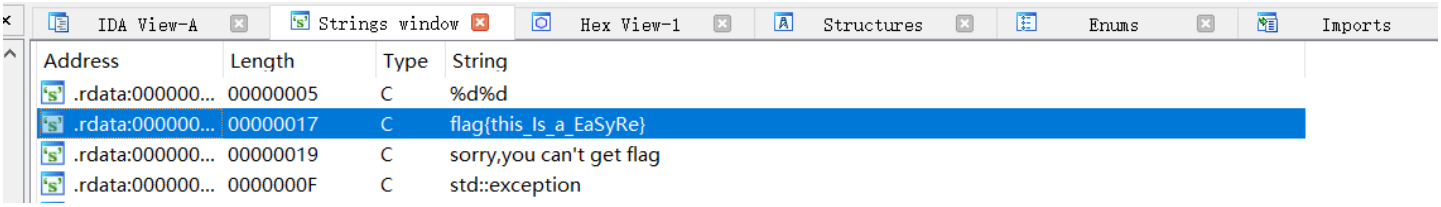
7 篇文章 0 订阅

订阅专栏

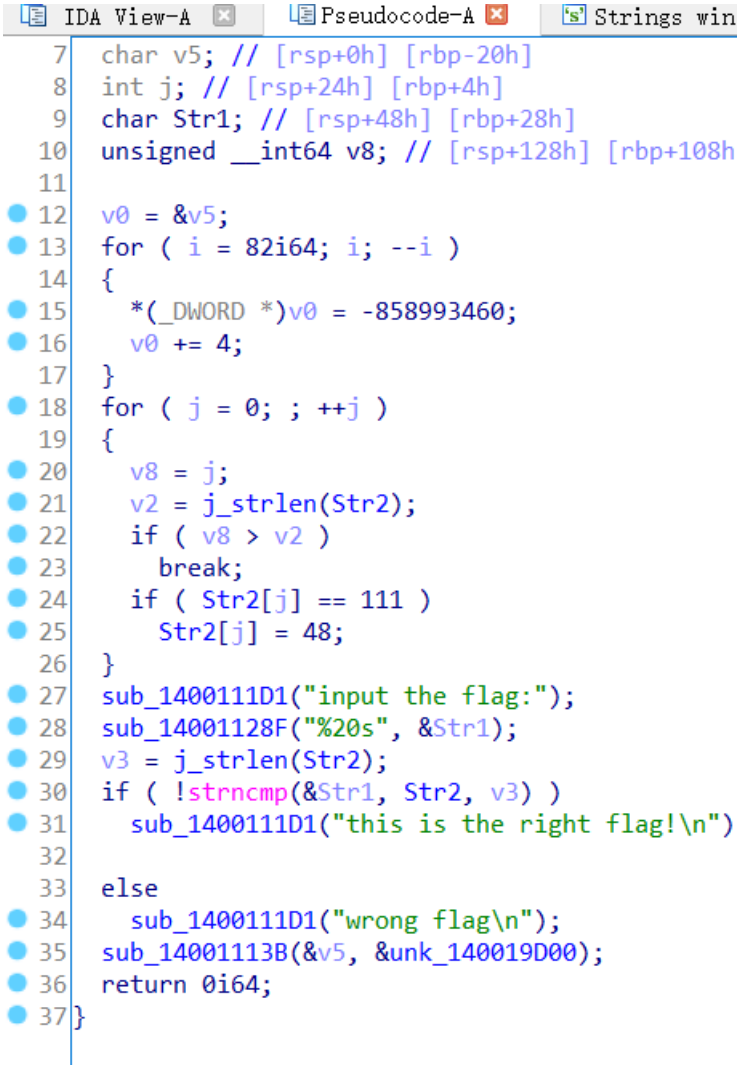
buuctf - re 刷题记录

1、easyre

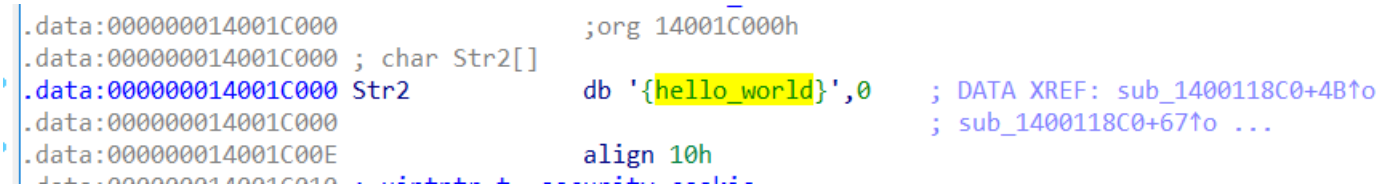
shift+f12



2、reverse1



双击跟进str2



24行到25行 o替换为0，flag{hell0_w0rld}，虽然很简单还是写写程序多熟练

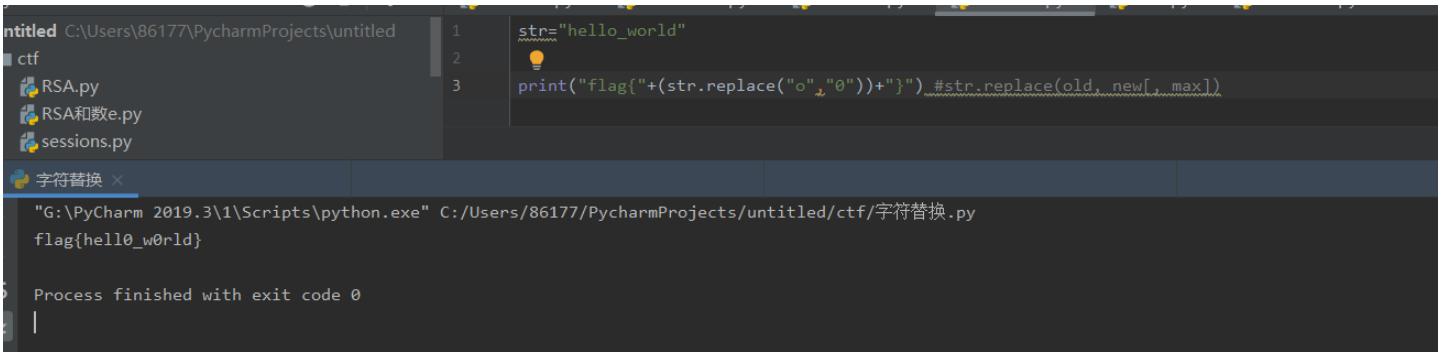
程序如下：

c语言：

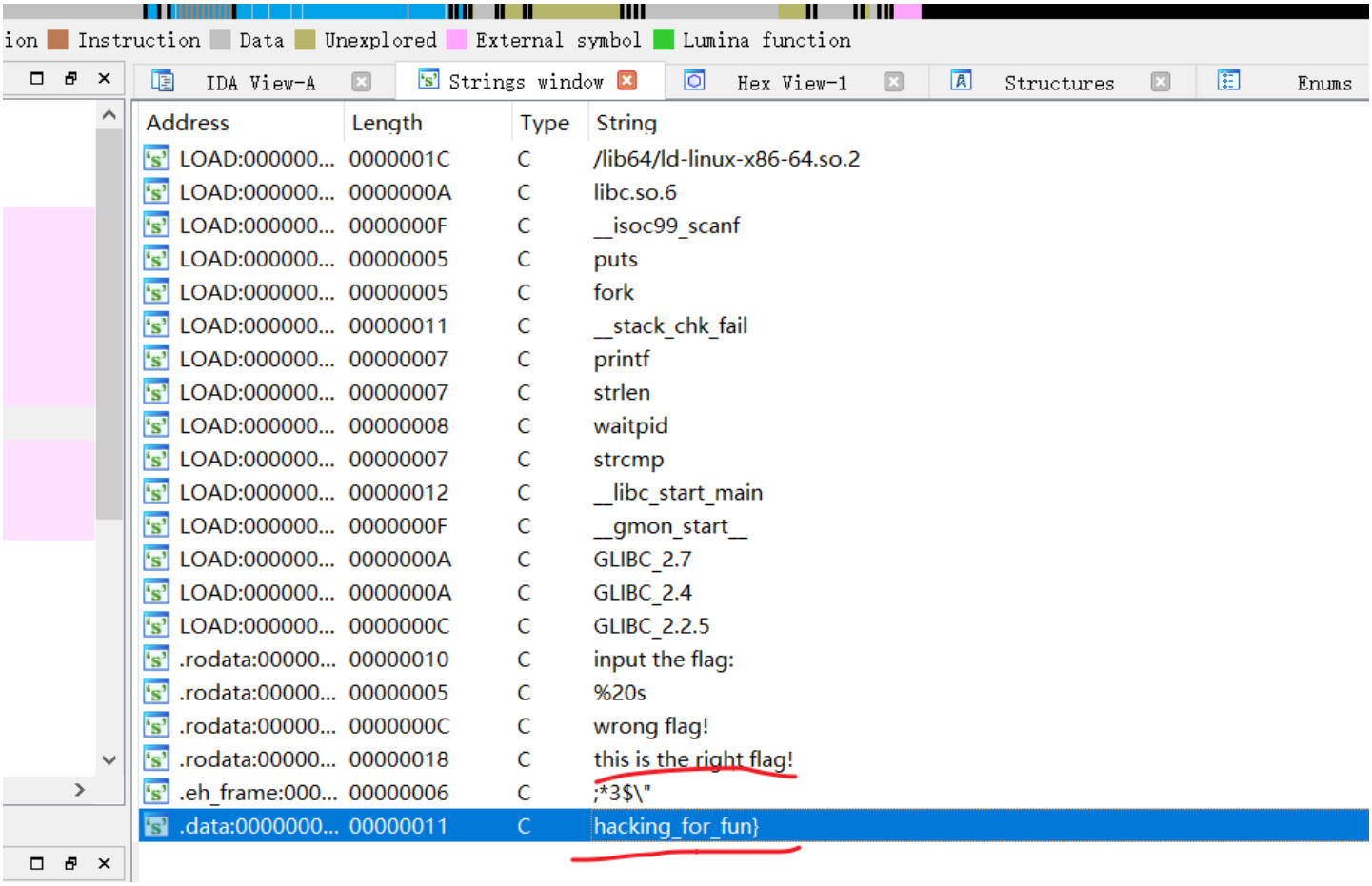
```c
#include<stdio.h>
#include<string.h>
int main()
{
  int v8,v2;
  char str2[]="hello_world";
  for (int j = 0; ; ++j )
  {
    v8 = j;
    v2 = strlen(str2);
    if ( v8 > v2 )
      break;
    if ( str2[j] == 'o' )
      str2[j] = '0';
  }
    printf("flag{%s}",str2);
}
```

```
C:\Users\86177\Desktop\buure\程序\reverse1.e
flag{hell0_w0rld}
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

python：



3、reverse2



简单查看下字符，疑似后半段flag，跟进对应字符查看

关键代码：

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-oNfwtff8-1635742416948)(C:\Users\86177\AppData\Roaming\Typora\typora-user-images\image-20211029190456094.png)]

跟进flag

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-lcjGyqS5-1635742416950)(C:\Users\86177\AppData\Roaming\Typora\typora-user-images\image-20211029190749227.png)]

看代码应该也是替换字符

模仿反编译的代码写下程序

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-swsQGaV1-1635742416952)(C:\Users\86177\AppData\Roaming\Typora\typora-user-images\image-20211029191358828.png)]

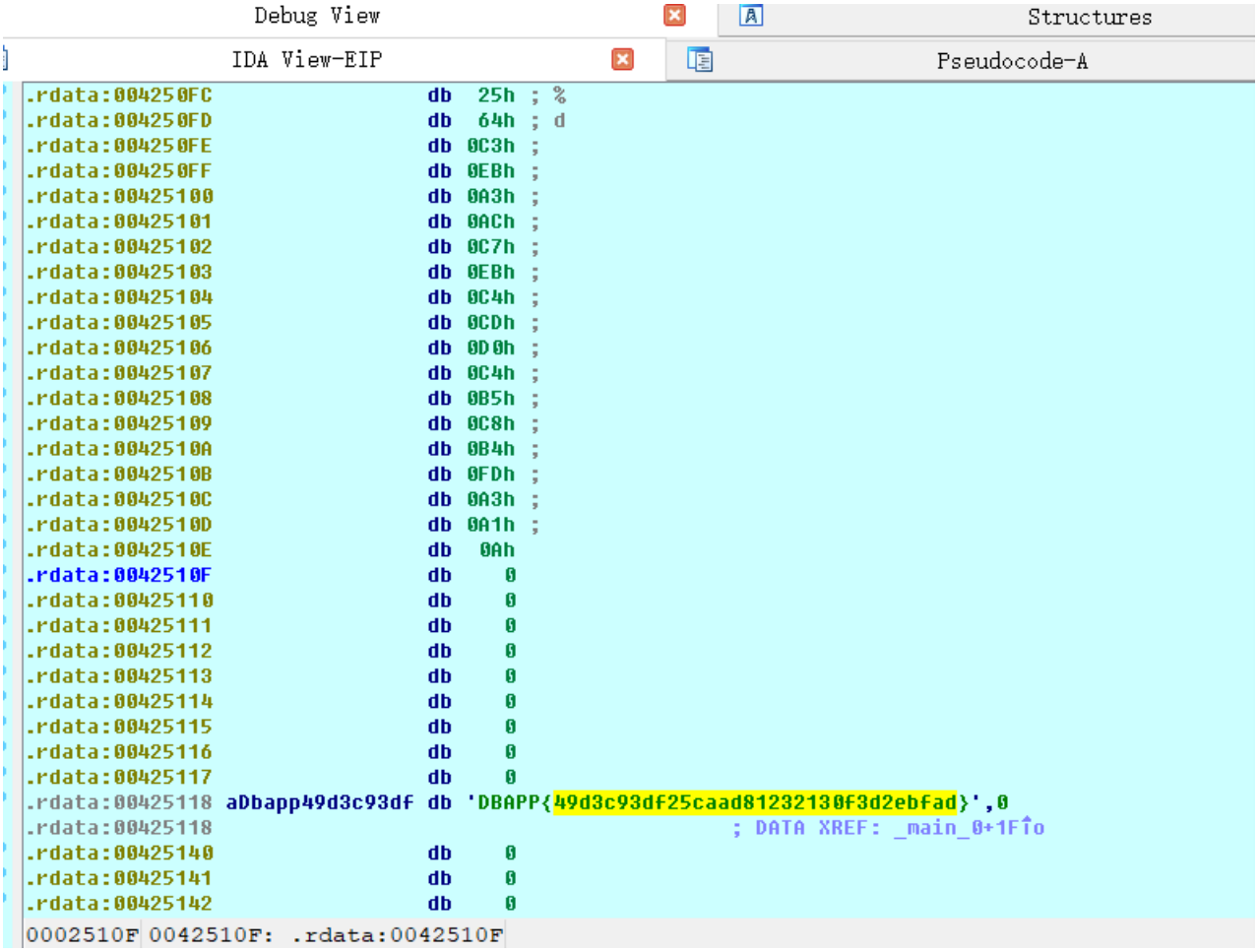
这两题都是简单的替换，只是程序一个是windows的exe，另一个是linux的可执行文件

python代码更简单，可以多学习学习，方便很多

```
str="hacking_for_fun}"

str2=str.replace("i","1")
print("flag{"+(str2.replace("r","1"))) #str.replace(old, new[, max])
```

4、内涵的软件

动态调试一下



但是其实这里就是输出的时候没有引入参数，值开始就已经给出了

5、新年快乐

查壳 upx

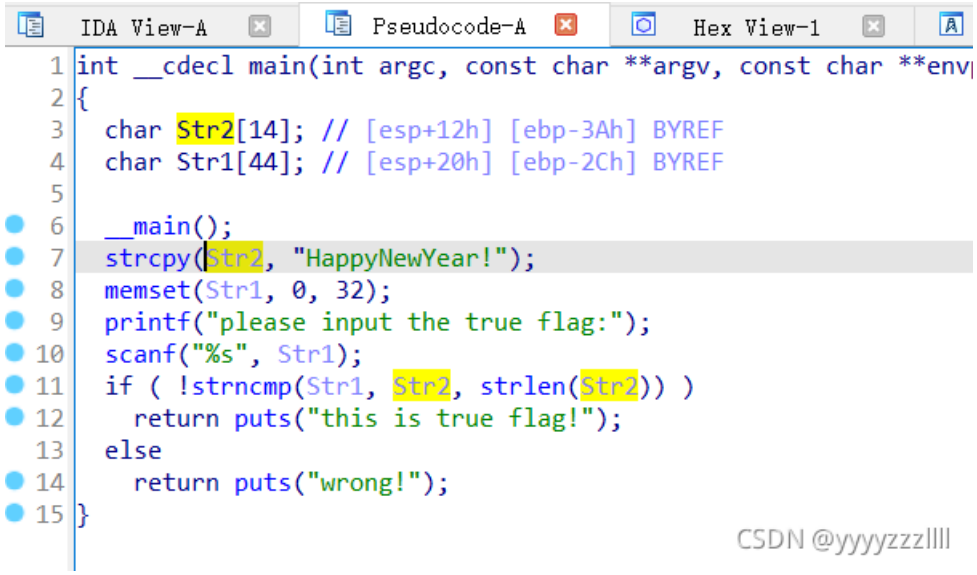
直接ida分析的话是打不开的，upx是压缩壳，程序基本上都被压缩加密的

脱壳

```
H:\CTFToolkit-v1.1_19.8.11\CTFToolkit-v1.1_19.8.11\逆向综合\upx-3.96-win64\upx-3.96-win64>upx.exe -d 新年快乐.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96w       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     27807 <-     21151   76.06%    win32/pe     新年快乐.exe

Unpacked 1 file.

H:\CTFToolkit-v1.1_19.8.11\CTFToolkit-v1.1_19.8.11\逆向综合\upx-3.96-win64\upx-3.96-win64>
```

CSDN @yyyyzzzlllll

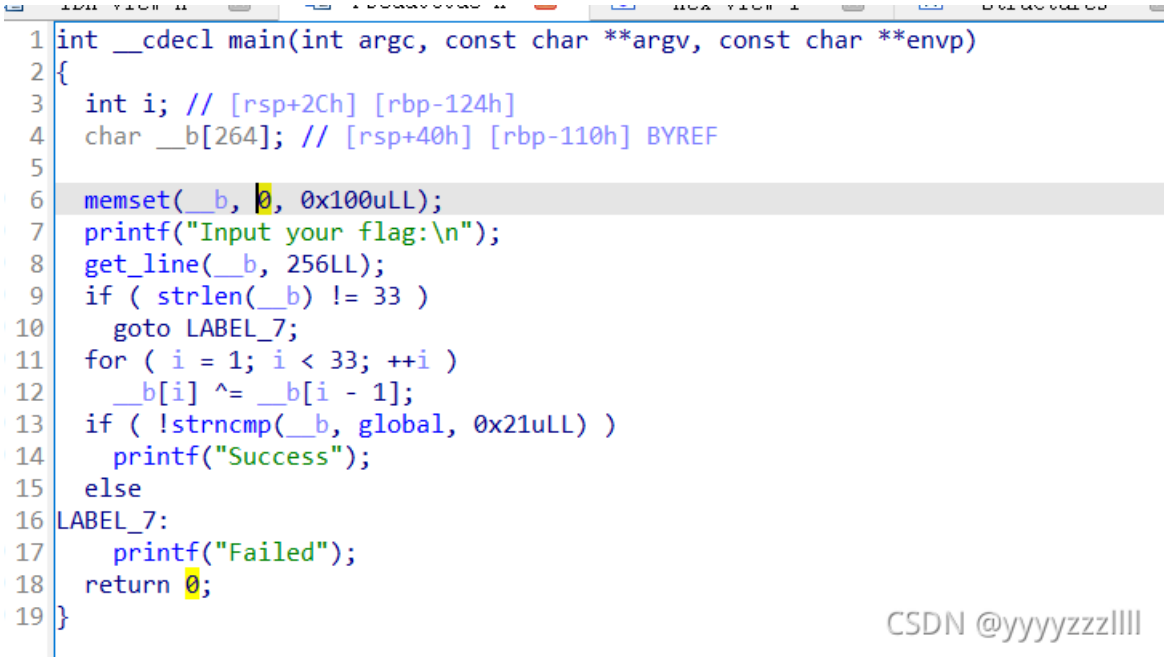
主函数



flag就是flag{HappyNewYear!}

6、xor

定位到关键代码



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+2Ch] [rbp-124h]
  char __b[264]; // [rsp+40h] [rbp-110h] BYREF

  memset(__b, 0, 0x100uLL);
  printf("Input your flag:\n");
  get_line(__b, 256LL);
  if ( strlen(__b) != 33 )
    goto LABEL_7;
  for ( i = 1; i < 33; ++i )
    __b[i] ^= __b[i - 1];
  if ( !strncmp(__b, global, 0x21uLL) )
    printf("Success");
  else
LABEL_7:
    printf("Failed");
  return 0;
}
```


找到数据

shift +E 导出数据

unsigned char ida_chars[] =
{
0x66, 0x0A, 0x6B, 0x0C, 0x77, 0x26, 0x4F, 0x2E, 0x40, 0x11,
0x78, 0x0D, 0x5A, 0x3B, 0x55, 0x11, 0x70, 0x19, 0x46, 0x1F,
0x76, 0x22, 0x4D, 0x23, 0x44, 0x0E, 0x67, 0x06, 0x68, 0x0F,
0x47, 0x32, 0x4F, 0x00
};

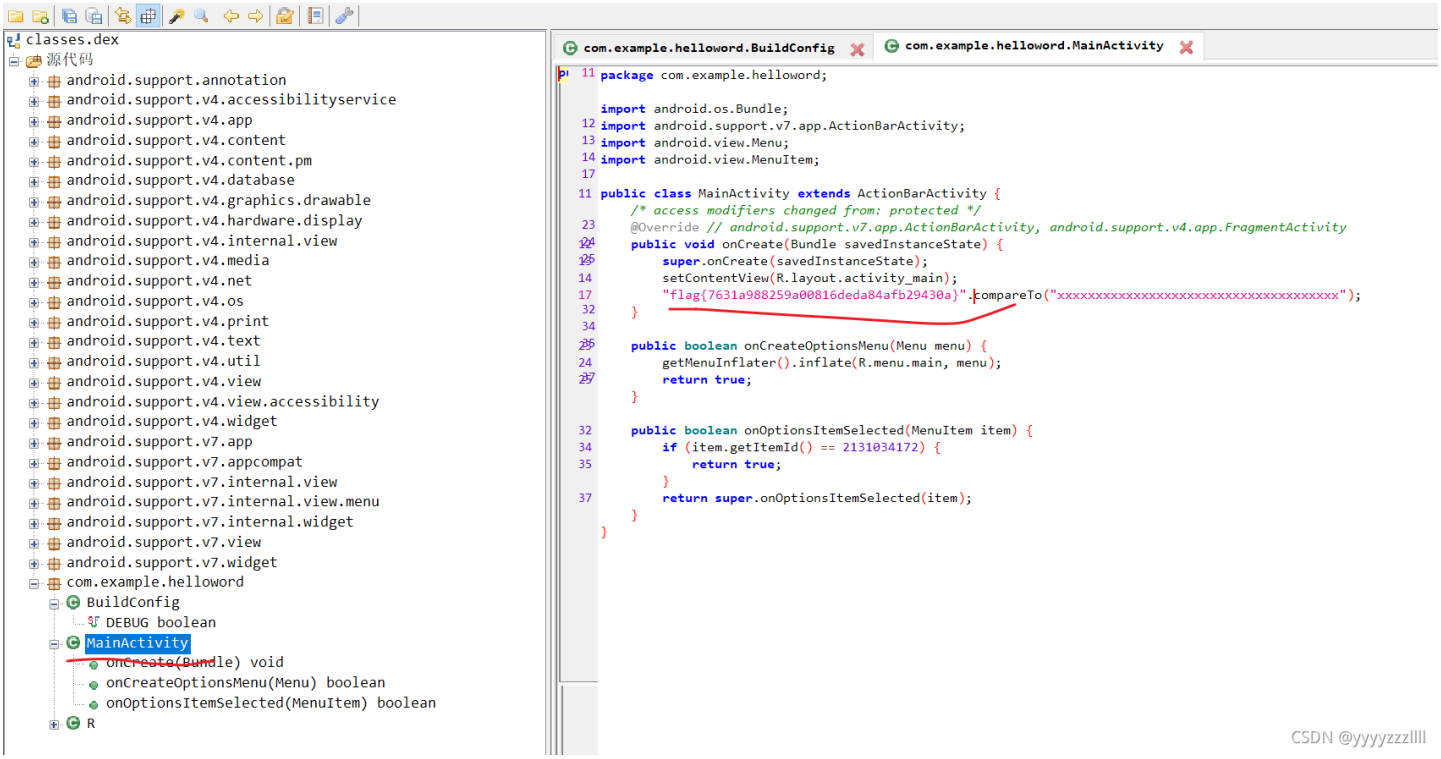
与前一位异或

```
data = [0x66, 0x0A, 0x6B, 0x0C, 0x77, 0x26, 0x4F, 0x2E, 0x40, 0x11,
        0x78, 0x0D, 0x5A, 0x3B, 0x55, 0x11, 0x70, 0x19, 0x46, 0x1F,
        0x76, 0x22, 0x4D, 0x23, 0x44, 0x0E, 0x67, 0x06, 0x68, 0x0F,
        0x47, 0x32, 0x4F, 0x00]
flag = chr(data[0])
for i in range(len(data)):
    flag += chr(data[i]^data[i-1])

print(flag)
```

7、helloword

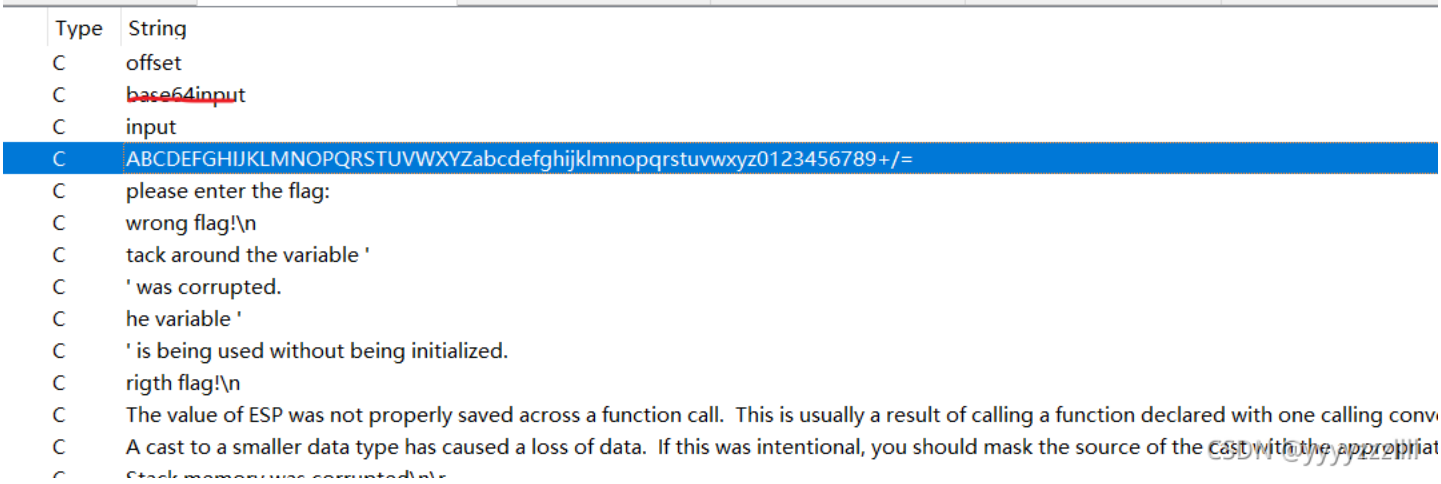
解压，反编译dex文件

main函数直接找到



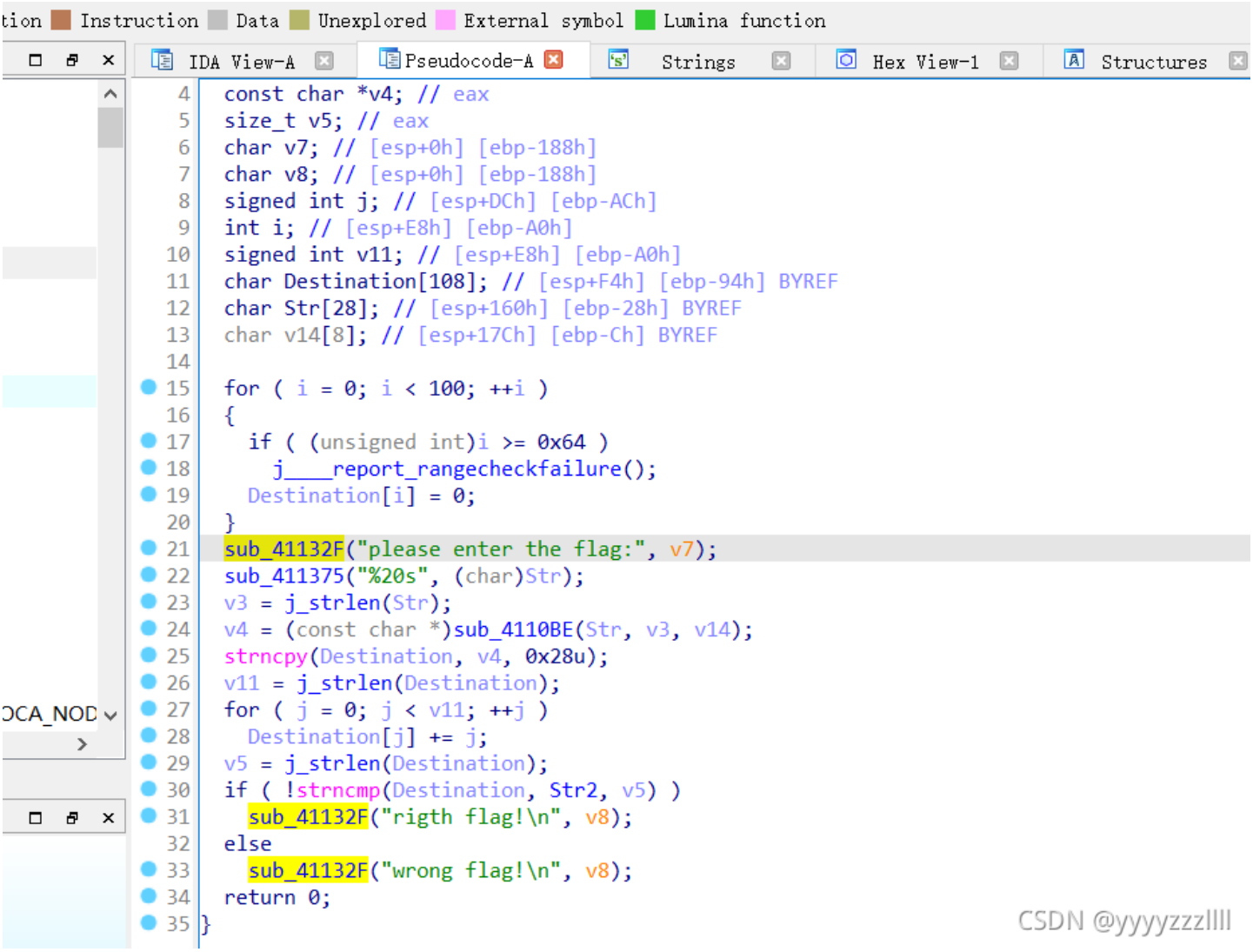
8、reverse3

无壳，32位 ,ida分析，简单看一眼字符，很可能是base64的简单加密



Type	String
C	offset
C	base64input
C	input
C	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=
C	please enter the flag:
C	wrong flag!\n
C	tack around the variable '
C	' was corrupted.
C	he variable '
C	' is being used without being initialized.
C	rigth flag!\n
C	The value of ESP was not properly saved across a function call. This is usually a result of calling a function declared with one calling conv
C	A cast to a smaller data type has caused a loss of data. If this was intentional, you should mask the source of the cast with the appropriat

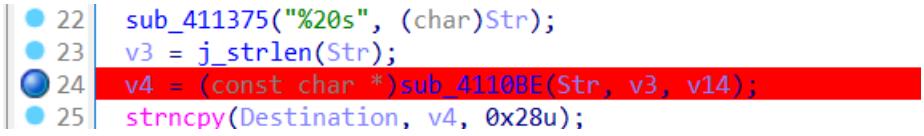
定位到主函数



可以适当修改来增加代码的可读性

输入的字符经过一个过程与str2比较，找到str2：e3niflH9b_C@n@dH

24行经历了一个变换

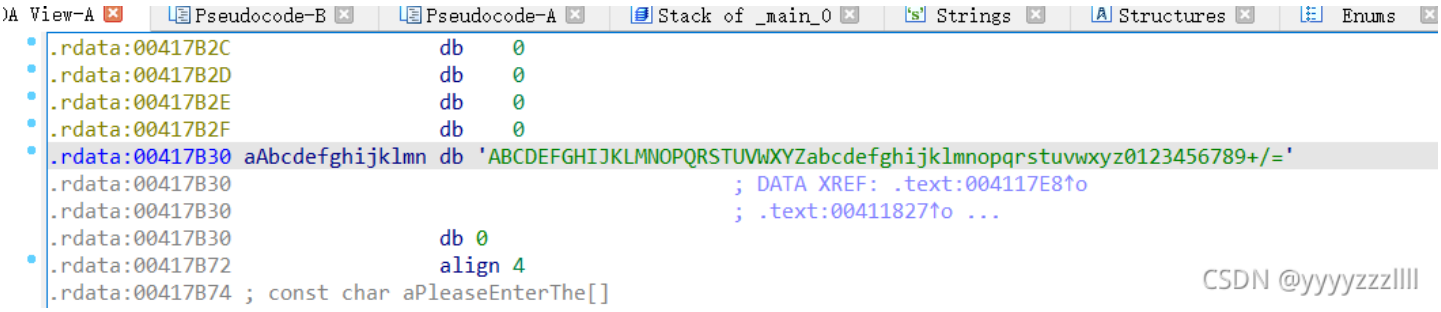


定位到这个加密函数

```
Instruction  Data  Unexplored  External symbol  Lumina function
IDA View-A | Pseudocode-B | Pseudocode-A | Stack of _main_0 | Strings | Structures | Enums | Imports | Exports
28  while ( v11 > 0 )
29  {
30    byte_41A144[2] = 0;
31    byte_41A144[1] = 0;
32    byte_41A144[0] = 0;
33    for ( i = 0; i < 3 && v11 >= 1; ++i )
34    {
35      byte_41A144[i] = *v13;
36      --v11;
37      ++v13;
38    }
39    if ( !i )
40      break;
41    switch ( i )
42    {
43      case 1:
44        *((_BYTE *)v12 + v4) = aAbcdefghijklmn[(int)(unsigned __int8)byte_41A144[0] >> 2];
45        v5 = v4 + 1;
46        *((_BYTE *)v12 + v5) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | (16 * (byte_41A144[0] & 3))];
47        *((_BYTE *)v12 + ++v5) = aAbcdefghijklmn[64];
48        *((_BYTE *)v12 + ++v5) = aAbcdefghijklmn[64];
49        v4 = v5 + 1;
50        break;
51      case 2:
52        *((_BYTE *)v12 + v4) = aAbcdefghijklmn[(int)(unsigned __int8)byte_41A144[0] >> 2];
53        v6 = v4 + 1;
54        *((_BYTE *)v12 + v6) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | (16 * (byte_41A144[0] & 3))];
55        *((_BYTE *)v12 + ++v6) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | (4 * (byte_41A144[1] & 0xF))];
56        *((_BYTE *)v12 + ++v6) = aAbcdefghijklmn[64];
57        v4 = v6 + 1;
58        break;
```


代码也看不懂，但是开头查看字符那里，有点像是base64加密

继续跟进一下



```
A View-A | Pseudocode-B | Pseudocode-A | Stack of _main_0 | Strings | Structures | Enums
.rdata:00417B2C                 db    0
.rdata:00417B2D                 db    0
.rdata:00417B2E                 db    0
.rdata:00417B2F                 db    0
.rdata:00417B30 aAbcdefghijklmn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='
.rdata:00417B30                                 ; DATA XREF: .text:004117E8↑o
.rdata:00417B30                                 ; .text:00411827↑o ...
.rdata:00417B30                 db 0
.rdata:00417B72                 align 4
.rdata:00417B74 ; const char aPleaseEnterThe[]
```


大概率是base64了

str2 还经过了28行这里的移位，先还原回来，再进行解密即可，直接解密base64也是解不了的，@这种字符解不了

解密脚本

```
import base64
str = "e3nifIH9b_C@n@dH"

flag =''
for i in range(len(str)):
    flag+=chr(ord(str[i])-i)

data = base64.b64decode(flag)
print(data)

#b'{i_l0ve_you}'
```

9、不一样的flag

定位到关键代码

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  _BYTE v3[29]; // [esp+17h] [ebp-35h] BYREF
  int v4; // [esp+34h] [ebp-18h]
  int v5; // [esp+38h] [ebp-14h] BYREF
  int i; // [esp+3Ch] [ebp-10h]
  char v7[12]; // [esp+40h] [ebp-Ch] BYREF

  __main();
  v3[26] = 0;
  *(_WORD *)&v3[27] = 0;
  v4 = 0;
  strcpy(v3, "*111101000010100000101111#");
  while ( 1 )
  {
    puts("you can choose one action to execute");
    puts("1 up");
    puts("2 down");
    puts("3 left");
    printf("4 right\n:");
    scanf("%d", &v5);
    if ( v5 == 2 )
    {
      ++*(_DWORD *)&v3[25];
    }
    else if ( v5 > 2 )
    {
      if ( v5 == 3 )
      {
        --v4;
      }
      else
      {
        if ( v5 != 4 )
LABEL_13:
          exit(1);
        ++v4;
      }
    }
    else
    {
      if ( v5 != 1 )
        goto LABEL_13;
      --*(_DWORD *)&v3[25];
    }
    for ( i = 0; i <= 1; ++i )
    {
      if ( *(int *)&v3[4 * i + 25] < 0 || *(int *)&v3[4 * i + 25] > 4 )
        exit(1);
    }
    if ( v7[5 * *(_DWORD *)&v3[25] - 41 + v4] == 49 )
      exit(1);
    if ( v7[5 * *(_DWORD *)&v3[25] - 41 + v4] == 35 )
    {
      puts("\nok, the order you enter is the flag!");
      exit(0);
    }
  }
}
```


大概率是个迷宫题

从下面这段代码可以看出

```
for ( i = 0; i <= 1; ++i )
{
  if ( *(int *)&v3[4 * i + 25] < 0 || *(int *)&v3[4 * i + 25] > 4 )
    exit(1);
}
if ( v7[5 * *(_DWORD *)&v3[25] - ')' + v4] == '1' )
  exit(1);
if ( v7[5 * *(_DWORD *)&v3[25] - ')' + v4] == '#' )
{
  puts("\nok, the order you enter is the flag!");
  exit(0);
}
```


*1111

01000

01010

00010

1111#

构成如上这样一个迷宫，从*开始不能触碰1到#结束，然后输入1234分别代表上下左右

很了然了，直接输出

flag{222441144222}

10、SimpleRev

找到main函数，似乎没有什么用，大概就是输入一个值跳到一个关卡的意味

```
1  int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2  {
3    int v3; // eax
4    char v4; // [rsp+Fh] [rbp-1h]
5
6    while ( 1 )
7    {
8      while ( 1 )
9      {
10       printf("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ");
11       v4 = getchar();
12       if ( v4 != 'd' && v4 != 'D' )
13         break;
14       Decry();
15     }
16     if ( v4 == 'q' || v4 == 'Q' )
17       Exit("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ", argv);
18     puts("Input fault format!");
19     v3 = getchar();
20     putchar(v3);
21   }
22 }
```


关键代码应该是Decry() 定位到该函数，

将字符拼接好

```
unsigned __int64 Decry()
{
  char v1; // [rsp+Fh] [rbp-51h]
  int v2; // [rsp+10h] [rbp-50h]
  int v3; // [rsp+14h] [rbp-4Ch]
```

```
int i; // [rsp+18h] [rbp-48h]
int len; // [rsp+1Ch] [rbp-44h]
char src[8]; // [rsp+20h] [rbp-40h] BYREF
__int64 v7; // [rsp+28h] [rbp-38h]
int v8; // [rsp+30h] [rbp-30h]
__int64 v9[2]; // [rsp+40h] [rbp-20h] BYREF
int v10; // [rsp+50h] [rbp-10h]
unsigned __int64 v11; // [rsp+58h] [rbp-8h]

v11 = __readfsqword(0x28u);
*(_QWORD *)src = 'SLCDN';
v7 = 0LL;
v8 = 0;
v9[0] = 'wodah';
v9[1] = '\0';
v10 = 0;
text = (char *)join(key3, v9);                // text = killshadow
strcpy(key, key1);
strcat(key, src);                             // key1=ADSFKNDCLS
v2 = 0;
v3 = 0;
getchar();
len = strlen(key);
for ( i = 0; i < len; ++i )
{
  if ( key[v3 % len] > '@' && key[v3 % len] <= 'Z' )
    key[i] = key[v3 % len] + ' ';
  ++v3;
}
printf("Please input your flag:");
while ( 1 )
{
  v1 = getchar();
  if ( v1 == '\n' )
    break;
  if ( v1 == ' ' )
  {
    ++v2;
  }
  else
  {
    if ( v1 <= '`' || v1 > 'z' )
    {
      if ( v1 > '@' && v1 <= 'Z' )
      {
        str2[v2] = (v1 - '\'' - key[v3 % len] + 'a') % '\x1A' + 'a';
        ++v3;
      }
    }
    else
    {
      str2[v2] = (v1 - 39 - key[v3 % len] + 97) % 26 + 97;
      ++v3;
    }
    if ( !(v3 % len) )
      putchar(32);
    ++v2;
  }
}
if ( !strncmp(text, str2) )
```

```
  if ( !strcmp(text, str2) )
    puts("Congratulation!\n");
  else
    puts("Try again!\n");
  return __readfsqword(0x28u) ^ v11;
}
```

最终结果是要str2同text做比较，已经得到text = killshadow和key=ADSFKNDCLS

然后通过代码的变化推算出str2,大概逻辑就是将key转化成小写，输入v1 然后经过转化生成str2[]

反过来想，我们输入是是字母，而且是大写字母，如果转化后满足与对应的text[]相等就是我们要输入的字符

代码

python

```python
s = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

key = 'adsfkndcls'
text = 'killshadow'

flag =''
for i in range(len(text)):
    str2=text[i]
    for j in s:
        if str2 == chr((ord(j) - 39 - ord(key[i % len(key)])+97)%26+97):
            flag+=j
print('flag{'+flag+'}')
#flag{KLDQCUDFZO}
```

11、Java逆向解密

直接进行分析

defpackage.Reverse

```java
package defpackage;

import java.util.ArrayList;
import java.util.Scanner;

/* renamed from: Reverse  reason: default package */
public class Reverse {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Please input the flag : ");
        String str = s.next();
        System.out.println("Your input is : ");
        System.out.println(str);
        Encrypt(str.toCharArray());
    }

    public static void Encrypt(char[] arr) {
        int[] KEY;
        ArrayList<Integer> Resultlist = new ArrayList<>();
        for (char c : arr) {
            Resultlist.add(Integer.valueOf((c + '@') ^ 32));
        }
        ArrayList<Integer> KEYList = new ArrayList<>();
        for (int i : new int[]{180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65}) {
            KEYList.add(Integer.valueOf(i));
        }
        System.out.println("Result:");
        if (Resultlist.equals(KEYList)) {
            System.out.println("Congratulations! ");
        } else {
            System.err.println("Error! ");
        }
    }
}
```


python

```python
list=[180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65]

flag=''
for i in range(len(list)):
    flag+=chr(list[i]-ord('@')^32)

print('flag{'+flag+'}')
#flag{This_is_the_flag_!}
```

它这里是java，也用java写一写吧

```java
package main;

public class main {
    public static void main(String[] args)
    {
        int[] list = {180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65};
        String flag="";
        for(int i=0;i<list.length;i++){
            char x = (char)((list[i]-'@')^32);
            flag+=x;
        }
        System.out.println("flag{"+flag+"}");
    }
}
//flag{This_is_the_flag_!}
```

12、[GXYCTF2019]luck_guy

```
1  int welcome()
2  {
3    puts("Welcome to Cyber SWAT2019.");
4    puts("Designed by Solar,wish you can enjoy it and have fun.");
5    return puts("Good luck (^_^)");
6  }
```

输入数字

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int v4; // [rsp+14h] [rbp-Ch] BYREF
4    unsigned __int64 v5; // [rsp+18h] [rbp-8h]
5
6    v5 = __readfsqword(0x28u);
7    welcome(argc, argv, envp);
8    puts("__________________");
9    puts("try to patch me and find flag");
.0   v4 = 0;
.1   puts("please input a lucky number");
.2   __isoc99_scanf("%d", &v4);
.3   patch_me(v4);
.4   puts("OK,see you again");
.5   return 0;
.6 }
```


跟进patch_me(),如果数字为偶数，进入get_flag函数

```
1  int __fastcall patch_me(int a1)
2  {
3    if ( a1 % 2 == 1 )
4      return puts("just finished");
5    else
6      return get_flag();
7  }
```

跟进get_flag()函数

```
1  unsigned __int64 get_flag()
2  {
3    unsigned int v0; // eax
4    int i; // [rsp+4h] [rbp-3Ch]
5    int j; // [rsp+8h] [rbp-38h]
6    __int64 s; // [rsp+10h] [rbp-30h] BYREF
7    char v5; // [rsp+18h] [rbp-28h]
8    unsigned __int64 v6; // [rsp+38h] [rbp-8h]
9
10   v6 = __readfsqword(0x28u);
11   v0 = time(0LL);
12   srand(v0);
13   for ( i = 0; i <= 4; ++i )
14   {
15     switch ( rand() % 200 )
16     {
17       case 1:
18         puts("OK, it's flag:");
19         memset(&s, 0, 0x28uLL);
20         strcat((char *)&s, f1);
21         strcat((char *)&s, &f2);
22         printf("%s", (const char *)&s);
23         break;
24       case 2:
25         printf("Solar not like you");
26         break;
27       case 3:
28         printf("Solar want a girlfriend");
29         break;
30       case 4:
31         s = 0x7F666F6067756369LL;
32         v5 = 0;
```


大概就是这些关键代码。接着分析,20、21行，有两个参数，f1=GXY{do_not_

f2指向的字符是s这里，大概是将两者拼接起来，那么关键点就在31行这一段16进制字符了

```
        memset(&s, 0, 0x28uLL);
        strcat((char *)&s, f1);
        strcat((char *)&s, &f2);
        printf("%s", (const char *)&s);
        break;
      case 2:
        printf("Solar not like you");
        break;
      case 3:
        printf("Solar want a girlfriend");
        break;
      case 4:
        s = 0x7F666F6067756369LL;
        v5 = 0;
        strcat(&f2, (const char *)&s);
        break;
      case 5:
        for ( j = 0; j <= 7; ++j )
        {
          if ( j % 2 == 1 )
            *(&f2 + j) -= 2;
          else
            --*(&f2 + j);
        }
        break;
      default:
        puts("emmm,you can't find flag 23333");
        break;
    }
  }
  return __readfsqword(0x28u) ^ v6;
}
```

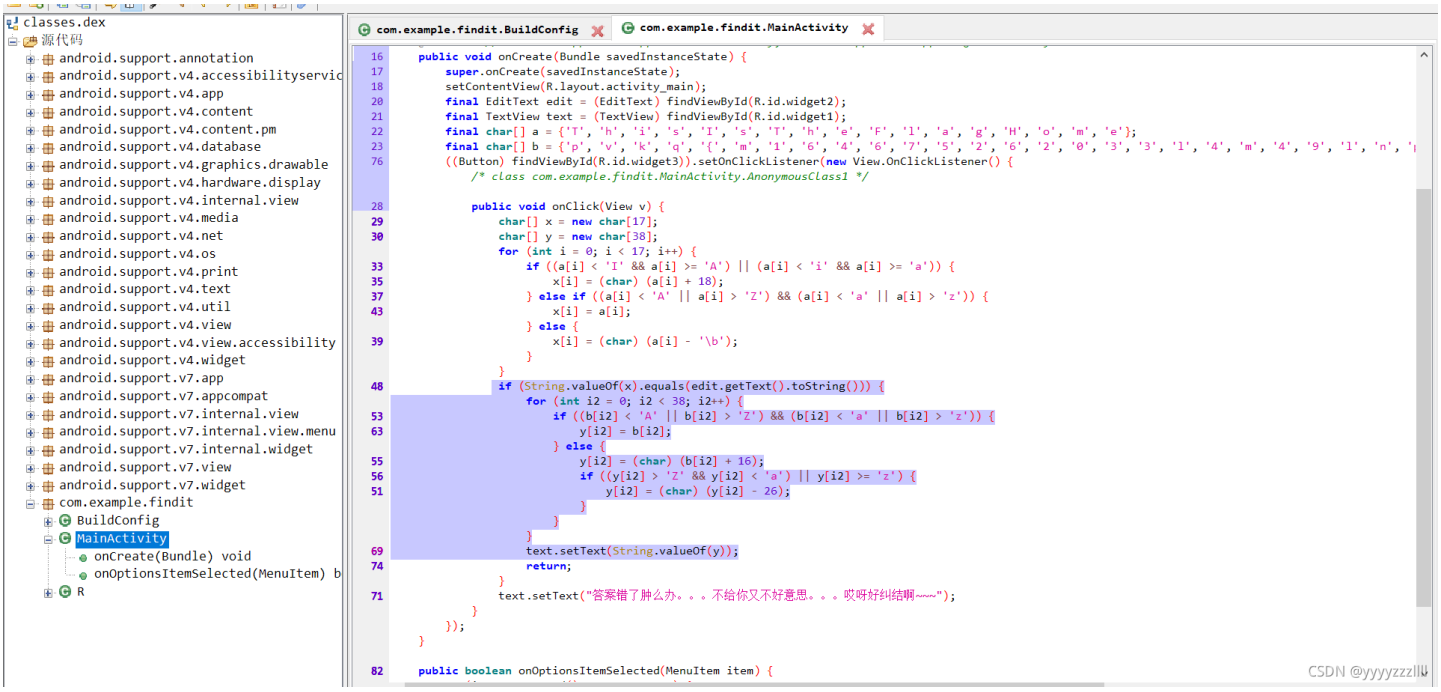

观察到case5：8位数的变换，将s的值转化一下就是：0x7F,0x66,0x6F,0x60,0x67,0x75,0x63,0x69

```
f1 = 'GXY{do_not_'
list=[0x7F,0x66,0x6F,0x60,0x67,0x75,0x63,0x69][::-1]

flag=''
for i in range (8):
    if i%2==1:
        s=chr(list[i]-2)
    else:
        s=chr(list[i]-1)
    flag+=s
print(f1+flag)
#GXY{do_not_hate_me}
```

代码有点长还没看，做实验去了

14、findit

下载是一个apk文件，直接解压找到class.dex，丢进jex反编译，找到主函数



直接定位到主函数

```
package com.example.findit;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {
    /* access modifiers changed from: protected */
    @Override // android.support.v7.app.ActionBarActivity, android.support.v4.app.FragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText edit = (EditText) findViewById(R.id.widget2);
        final TextView text = (TextView) findViewById(R.id.widget1);
        final char[] a = {'T', 'h', 'i', 's', 'I', 's', 'T', 'h', 'e', 'F', 'l', 'a', 'g', 'H', 'o', 'm', 'e'};
        final char[] b = {'p', 'v', 'k', 'q', '{', 'm', '1', '6', '4', '6', '7', '5', '2', '6', '2', '0', '3', '
3', 'l', '4', 'm', '4', '9', 'l', 'n', 'p', '7', 'p', '9', 'm', 'n', 'k', '2', '8', 'k', '7', '5', '}'};
        ((Button) findViewById(R.id.widget3)).setOnClickListener(new View.OnClickListener() {
            /* class com.example.findit.MainActivity.AnonymousClass1 */

            public void onClick(View v) {
                char[] x = new char[17];
                char[] y = new char[38];
                for (int i = 0; i < 17; i++) {
                    if ((a[i] < 'I' && a[i] >= 'A') || (a[i] < 'i' && a[i] >= 'a')) {
                        x[i] = (char) (a[i] + 18);
                    } else if ((a[i] < 'A' || a[i] > 'Z') && (a[i] < 'a' || a[i] > 'z')) {
                        x[i] = a[i];
                    } else {
                        x[i] = (char) (a[i] - '\b');
                    }
                }
                if (String.valueOf(x).equals(edit.getText().toString())) {
                    for (int i2 = 0; i2 < 38; i2++) {
```

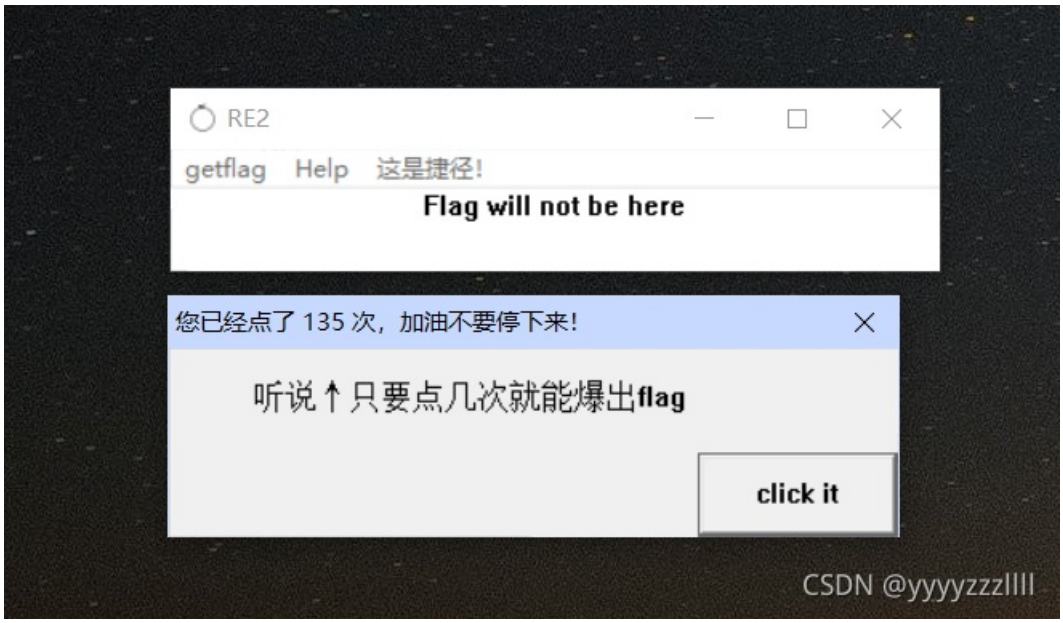
```
                for (int i2 = 0; i2 < 38; i2++) {
                    if ((b[i2] < 'A' || b[i2] > 'Z') && (b[i2] < 'a' || b[i2] > 'z')) {
                        y[i2] = b[i2];
                    } else {
                        y[i2] = (char) (b[i2] + 16);
                        if ((y[i2] > 'Z' && y[i2] < 'a') || y[i2] >= 'z') {
                            y[i2] = (char) (y[i2] - 26);
                        }
                    }
                }
                text.setText(String.valueOf(y));
                return;
            }
            text.setText("答案错了肿么办。。。不给你又不好意思。。。哎呀好纠结啊~~~");
        }
    });
}

public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == 2131034176) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
}
```

模仿代码写出解密

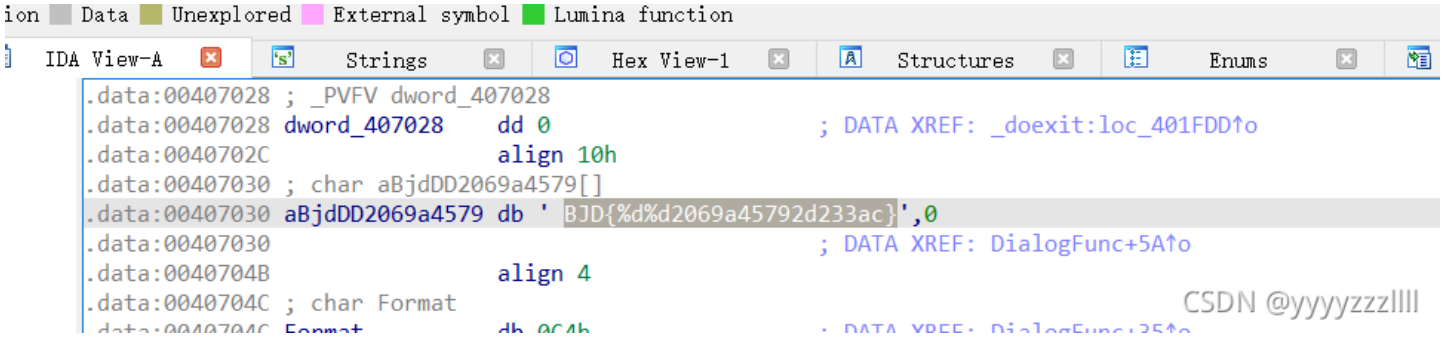
```
package main;

public class findit {
    public static void main(String[] args)
    {
        final char[] b = {'p', 'v', 'k', 'q', '{', 'm', '1', '6', '4', '6', '7', '5', '2', '6', '2', '0', '3', '
3', 'l', '4', 'm', '4', '9', 'l', 'n', 'p', '7', 'p', '9', 'm', 'n', 'k', '2', '8', 'k', '7', '5', '}'};

        char[] y = new char[38];
        for (int i2 = 0; i2 < 38; i2++) {
            if ((b[i2] < 'A' || b[i2] > 'Z') && (b[i2] < 'a' || b[i2] > 'z')) {
                y[i2] = b[i2];
            } else {
                y[i2] = (char) (b[i2] + 16);
                if ((y[i2] > 'Z' && y[i2] < 'a') || y[i2] >= 'z') {
                    y[i2] = (char) (y[i2] - 26);
                }
            }
        }
        for(int j=0;j<38;j++){
            System.out.print(y[j]);
        }

    }
}
//flag{c164675262033b4c49bdf7f9cda28a75}
```

15、[BJDCTF2020]JustRE

打开后一个这软件



ida看看，发现一个字符，但是%d%d这不知道是什么鬼，继续分析



大概是点击19999次出flag？？按照代码意思将19999和0填入%d%d BJD{1999902069a45792d233ac}

->flag{1999902069a45792d233ac}

```
INT_PTR __stdcall DialogFunc(HWND hWnd, UINT a2, WPARAM a3, LPARAM a4)
{
  CHAR String[100]; // [esp+0h] [ebp-64h] BYREF

  if ( a2 != 272 )
  {
    if ( a2 != 273 )
      return 0;
    if ( (_WORD)a3 != 1 && (_WORD)a3 != 2 )
    {
      sprintf(String, &Format, ++dword_4099F0);
      if ( dword_4099F0 == 19999 )
      {
        sprintf(String, " BJD{%d%d2069a45792d233ac}", 19999, 0);
        SetWindowTextA(hWnd, String);
        return 0;
      }
      SetWindowTextA(hWnd, String);
      return 0;
    }
    EndDialog(hWnd, (unsigned __int16)a3);
  }
  return 1;
}
```


16、简单注册器

和findit类似，直接找主函数

直接看验证部分就可以其实， 这种筛选部分直接可以不看，过滤不满足的字符然后跳转到retuen；

String xx = editview.getText().toString();

if (!(xx.length() == 32 && xx.charAt(31) == ‘a’ && xx.charAt(1) == ‘b’ && (xx.charAt(0) + xx.charAt(2)) - 48 == 56)) {
flag = 0;
}

```
package main;

public class 简单注册表 {
    public static void main(String[] args) {
        int flag = 1;
        if (flag == 1) {
            char[] x = "dd2940c04462b4dd7c450528835cca15".toCharArray();
            x[2] = (char) ((x[2] + x[3]) - 50);
            x[4] = (char) ((x[2] + x[5]) - 48);
            x[30] = (char) ((x[31] + x[9]) - 48);
            x[14] = (char) ((x[27] + x[28]) - 97);
            for (int i = 0; i < 16; i++) {
                char a = x[31 - i];
                x[31 - i] = x[i];
                x[i] = a;
            }
            System.out.print("flag{" + String.valueOf(x) + "}");
        }
    }
}
//flag{59acc538825054c7de4b26440c0999dd}
```

17、[GWCTF 2019]pyre

pyc文件，直接在线反编译成py，原理后面再学一下

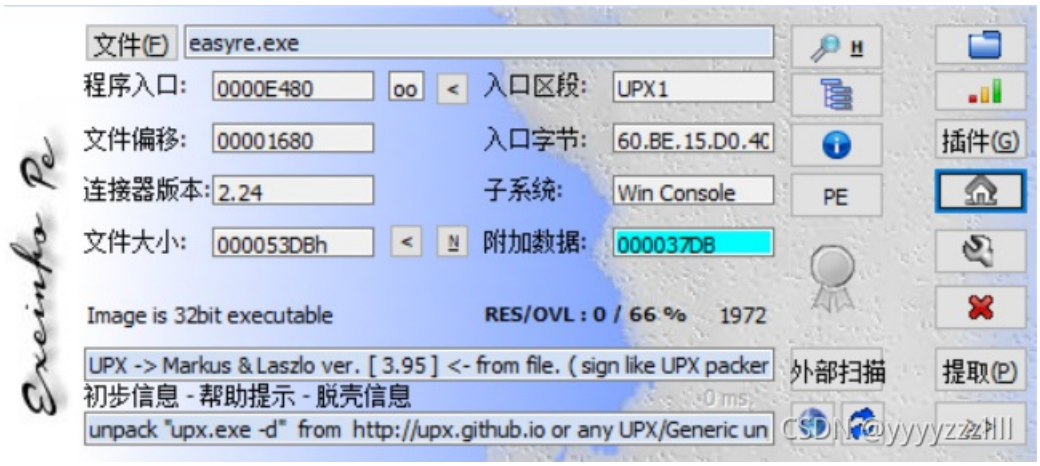
```
#!/usr/bin/env python
# visit https://tool.lu/pyc/ for more information
print 'Welcome to Re World!'
print 'Your input1 is your flag~'
l = len(input1)
for i in range(l):
    num = ((input1[i] + i) % 128 + 128) % 128
    code += num

for i in range(l - 1):
    code[i] = code[i] ^ code[i + 1]

print code
code = [
    '\x1f',
    '\x12',
    '\x1d',
    '(',
    '0',
    '4',
    '\x01',
    '\x06',
    '\x14',
    '4',
    ',',
    '\x1b',
    'U',
    '?',
    'o',
    '6',
    '*',
    ':',
    '\x01',
    'D',
    ';',
    '%',
    '\x13']
```

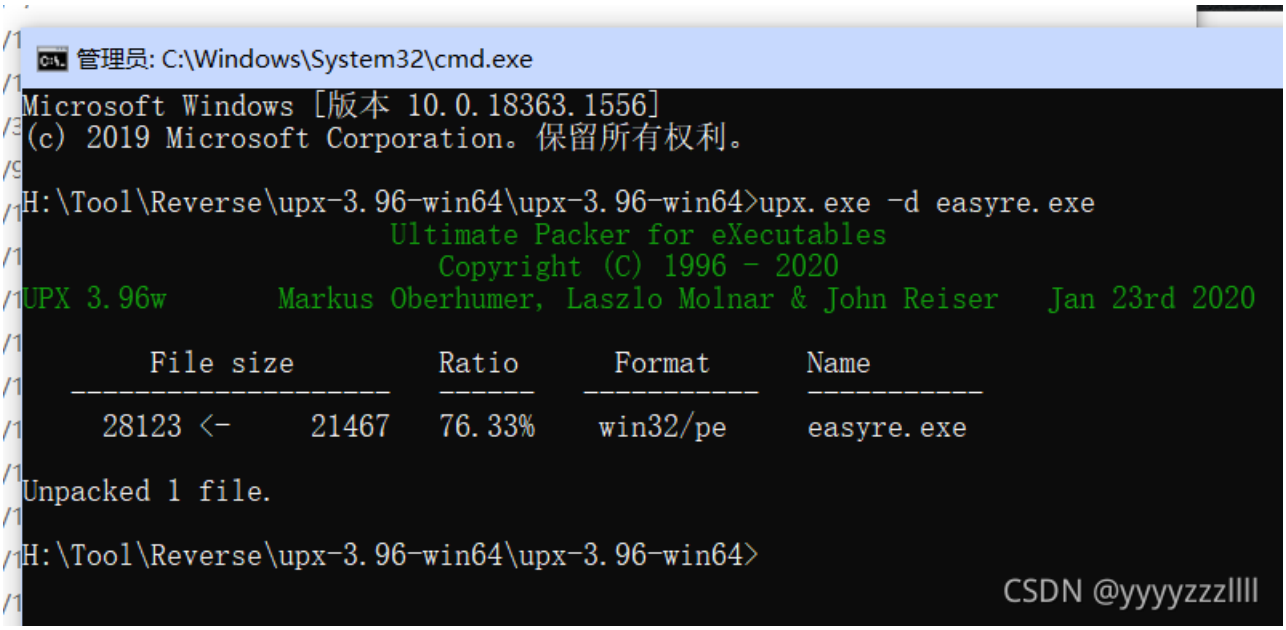
解密：

```
code = ['\x1f','\x12','\x1d','(','0','4','\x01','\x06','\x14','4',',','\x1b','U','?','o','6','*',':','\x01','D',
';','%','\x13']

l = len(code)
flag=''
for i in range(l-2,-1,-1):
    code[i]=chr(ord(code[i])^ord(code[i+1]))

for i in range(len(code)):
    flag+=chr((ord(code[i])-i)%128)

print(flag)
# GWHT{Just_Re_1s_Ha66y!}
```

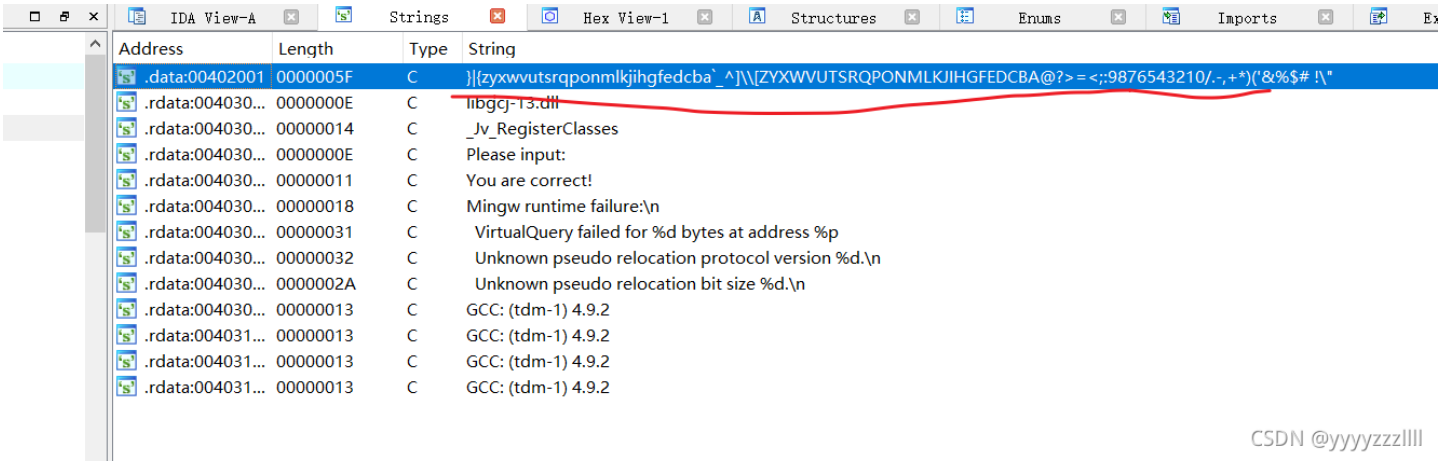
18、[ACTF新生赛2020]easyre

UPX壳，脱去





然后IDA32打开，貌似有个什么加密

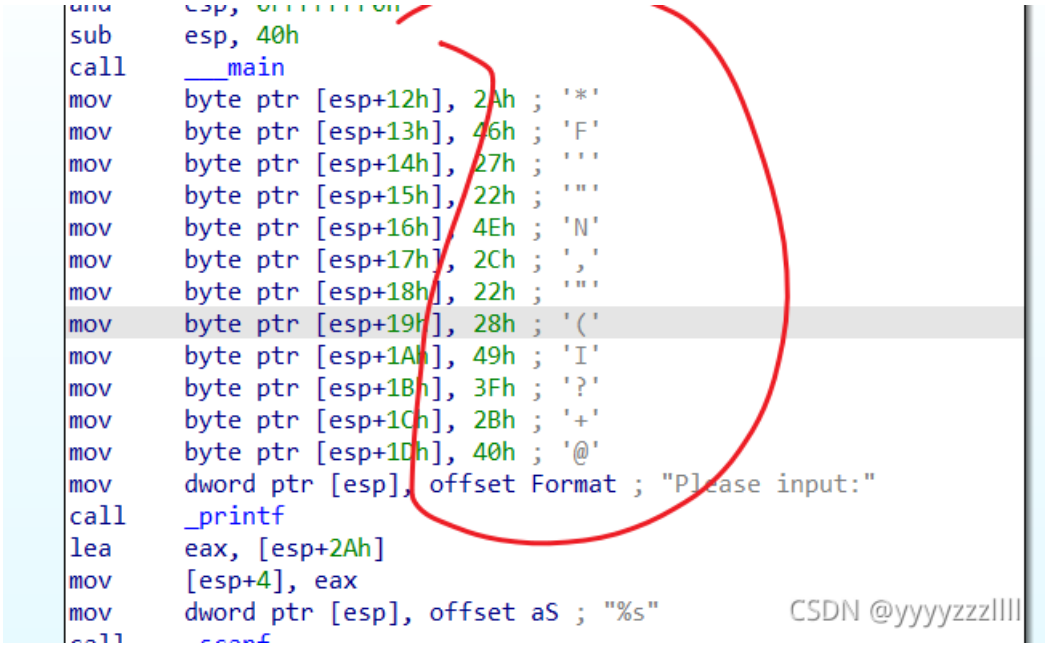


定位到主函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4[12]; // [esp+12h] [ebp-2Eh] BYREF
  int v5[3]; // [esp+1Eh] [ebp-22h]
  char v6[5]; // [esp+2Ah] [ebp-16h] BYREF
  int v7; // [esp+2Fh] [ebp-11h]
  int v8; // [esp+33h] [ebp-Dh]
  int v9; // [esp+37h] [ebp-9h]
  char v10; // [esp+3Bh] [ebp-5h]
  int i; // [esp+3Ch] [ebp-4h]

  __main();
  qmemcpy(v4, "*F'\"N,\"(I?+@", sizeof(v4));
  printf("Please input:");
  scanf("%s", v6);
  if ( v6[0] != 65 || v6[1] != 67 || v6[2] != 84 || v6[3] != 70 || v6[4] != 123 || v10 != 125 )
    return 0;
  v5[0] = v7;
  v5[1] = v8;
  v5[2] = v9;
  for ( i = 0; i <= 11; ++i )
  {
    if ( v4[i] != _data_start__[*((char *)v5 + i) - 1] )    //v4[i]得与_data_start处理后相等
      return 0;
  }
  printf("You are correct!");
  return 0;
}
```

需要是以ASCTF{}包含一个文件，大概中间是12个字符，与"}|{zyxwvutsrqponmlkjihgfedcba`_^]
[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>="进行加密

找到那些字符



第一次做这种，抄了个脚本

```
v4 = [42,70,39,34,78,44,34,40,73,63,43,64]
string = chr(0x7E)+"}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)(" + ch
r(0x27) + '&%$# !"'
flag=""

for i in v4:
    for j in range(1,len(string)):
        if i == ord(string[j]):
            flag+=chr(j+1)

print ("flag{"+flag+"}")
#flag{U9X_1S_W6@T?}
```