

buu逆向刷题（三）

原创

北风~ 于 2020-09-29 22:23:10 发布 6061 收藏 1

分类专栏: [逆向与保护](#)

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

本文链接: https://blog.csdn.net/weixin_45055269/article/details/108814610

版权



[逆向与保护](#) 专栏收录该内容

65 篇文章 4 订阅

订阅专栏

1. [BJDCTF2020]JustRE

点击次数, IDA找到逻辑, OD直接跳到相应位置, 改掉比较的值, 获取flag。

• 00401380	50	push eax	
• 00401381	8D4424 04	lea eax,dword ptr ss:[esp+4]	[esp+4]: "BJD{%d%d2069a45792d233ac
• 00401385	68 4C704000	push [bjdctf2020]justre.40704C	40704c: "您已经点了 %d 次, 加油不要停下:
• 0040138A	50	push eax	
• 0040138B	E8 80000000	call [bjdctf2020]justre.401410	
• 00401390	A1 F0994000	mov eax,dword ptr ds:[4099F0]	
• 00401395	83C4 0C	add esp,C	
• 00401398	3D 1F4E0000	cmp eax,4E1F	
• 0040139D	75 31	jne [bjdctf2020]justre.4013B0	
• 0040139F	6A 00	push 0	
• 004013A1	68 1F4E0000	push 4E1F	
• 004013A6	8D4C24 08	lea ecx,dword ptr ss:[esp+8]	
• 004013AA	68 30704000	push [bjdctf2020]justre.407030	407030: "BJD{%d%d2069a45792d233ac}
• 004013AF	51	push ecx	
• 004013B0	E8 5B000000	call [bjdctf2020]justre.401410	
• 004013B5	8B4424 78	mov eax,dword ptr ss:[esp+78]	
• 004013B9	83C4 10	add esp,10	
• 004013BC	8D5424 00	lea edx,dword ptr ss:[esp]	[esp]: "BJD{1999902069a45792d233ac
• 004013C0	52	push edx	edx: ""
• 004013C1	50	push eax	https://blog.csdn.net/weixin_45055269

2.Youngter-drive

upx壳的反调试, upx脱壳, IDA可以打开, 但OD不行。

IDA打开

```
1 int main_0()
2 {
3     HANDLE v1; // [esp+D0h] [ebp-14h]
4     HANDLE hObject; // [esp+DCh] [ebp-8h]
5
6     sub_4110FF(); // 输入字符
7     ::hObject = CreateMutexW(0, 0, 0);
8     j_strcpy(Dest, Source);
9     hObject = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, 0); // 创建两个线程 对source字符替换, dword_418008减1
10    v1 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_41119F, 0, 0, 0); // dword_418008减1
11    CloseHandle(hObject);
12    CloseHandle(v1);
13    while ( dword_418008 != -1 )
14        ;
15    sub_411190(); // source字符比较
16    CloseHandle(::hObject);
17    return 0;
18 }
```

https://blog.csdn.net/weixin_45055269

逻辑清晰, **奇偶位置字符操作**创建两个线程, 一个线程对source字符串操作, 同时dword_418008减1 (此变量用于计数), 另一个线程dword_418008减1, 也就是说计数变量每次会减2, 接着考虑到计数变量开始是29, 所以source字符串里只有奇数位置的字符会被操作, 偶数字符保持不变。同时计数变量从29开始, 即source字符串30个字符。

sub_411190()函数只比较前29个字符。**比较的字符与原本字符的长度不同**

看看对source字符串做了如何的操作, StartAddress函数往里点, 点到sub_41112C函数反编译不了, 报错(411A04: positive sp value has been found), **IDA 调整栈帧** 参考链接

- 1) 打开Option->General->stack pointer勾选
- 2) 找到红色区域
- 3) ALT+K盘它
- 4) 不知道该多少就试一试, 只要最下面的-04改成0以上就可F5。

修改后

```
• .text:004119EA 0D8      cmp     esi, esp
• .text:004119EC 0D8      call   j__RTC_CheckEsp
.text:004119F1
.text:004119F1      loc_4119F1:                                ; CODE XREF: sub_41194
• .text:004119F1 0D8      pop     edi
• .text:004119F2 0D4      pop     esi
• .text:004119F3 0D0      pop     ebx
• .text:004119F4 0CC      add     esp, 0CCh
• .text:004119FA 000      cmp     ebp, esp
• .text:004119FC 000      call   j__RTC_CheckEsp
• .text:00411A01 000      mov     esp, ebp
• .text:00411A03 000      pop     ebp
• .text:00411A04 000      retn
.text:00411A04      sub_411940 endp ; sp-analysis failed
.text:00411A04
```

https://blog.csdn.net/weixin_45055269

脚本:

```

s = 'QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm' #off_418000
d = 'TOiZiZtOrYaToUwPnToBsOaOapsyS' #off_418004
f = ''
for i in range(len(d)):
    if i%2 == 0:
        f = f + d[i]
    else:
        if(d[i].isupper()):
            f = f + chr(s.find(d[i])+96)
        else:
            f = f + chr(s.find(d[i])+38)
print(f)
#ThisisthreadofwindowshahaIsES

```

最后一个字符试了试发现是E, ThisisthreadofwindowshahalsESE提交

3.[2019红帽杯]easyRE

搜索字符串, 找到关键函数sub_4009C6, 一个异或, 一个10次的base64加密

1) 异或脚本

```

tmp=[73,111,100,108,62,81,110,98,40,111,99,121,127,121,46,105,127,100,96,51,119,125,119,101,107,57,123,105,121,6
! [在这里插入图片描述](https://img-blog.csdnimg.cn/20200927105642121.png?x-oss-process=image/watermark,type_ZmFuZ3p
oZW5naGVpdGk,shadow_10,text_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlbXhpb180NTA1NTI2OQ==,size_16,color_FFFFFFFF,t_70#pic_c
enter)
1,126,121,76,64,69,67]
flag=[0 for i in range(36)]
for i in range(36):
    flag[i]=tmp[i]^i
print(''.join(map(chr,flag)))
#Info:The first four chars are `flag

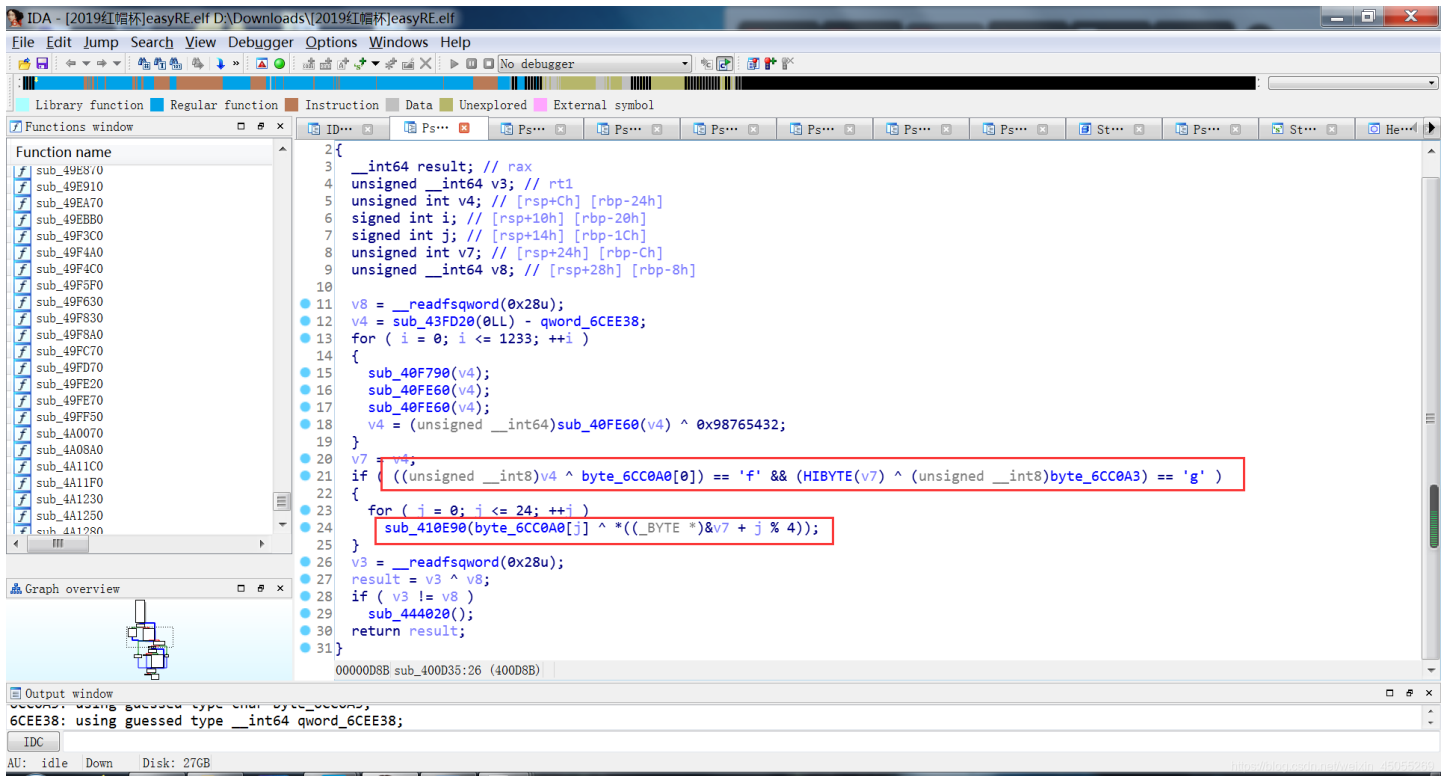
```

2) base64 10次解密后得<https://bbs.pediy.com/thread-254172.htm>, 这个网站没有flag。

没有flag, 从哪里找, 有些函数在不知道的地方执行。base64 10次解密的字符串, 下面有可疑字符

```
.data:00000000006CC08E db 0
.data:00000000006CC08F db 0
.data:00000000006CC090 off_6CC090 dq offset aVm0wd2vhuxhtwg
.data:00000000006CC090 ; DATA XREF: sub_4009C6+31B↑r
.data:00000000006CC090 ; "Vm0wd2VHUXhTWGhpUm1SWVYwZDRWV113Wkc5WFJ"...
.data:00000000006CC098 align 20h
.data:00000000006CC0A0 ; char byte_6CC0A0[3]
.data:00000000006CC0A0 byte_6CC0A0 db 40h ; DATA XREF: sub_400D35+95↑r
.data:00000000006CC0A0 ; sub_400D35+C1↑r
.data:00000000006CC0A1 db 35h ; 5
.data:00000000006CC0A2 db 20h
.data:00000000006CC0A3 byte_6CC0A3 db 56h ; DATA XREF: sub_400D35+A6↑r
.data:00000000006CC0A4 db 5Dh ; ]
.data:00000000006CC0A5 db 18h
.data:00000000006CC0A6 db 22h ; "
.data:00000000006CC0A7 db 45h ; E
.data:00000000006CC0A8 db 17h
.data:00000000006CC0A9 db 2Fh ; /
.data:00000000006CC0AA db 24h ; $
.data:00000000006CC0AB db 6Eh ; n
.data:00000000006CC0AC db 62h ; b
.data:00000000006CC0AD db 3Ch ; <
.data:00000000006CC0AE db 27h ; '
.data:00000000006CC0AF db 54h ; T
.data:00000000006CC0B0 db 48h ; H
.data:00000000006CC0B1 db 6Ch ; l
.data:00000000006CC0B2 db 24h ; $
.data:00000000006CC0B3 db 6Eh ; n
000CC0A5 00000000006CC0A5: .data:00000000006CC0A5 (Synchronized with Hex View-1) https://blog.csdn.net/weixin_45055269
```

交叉引用过去可以看到



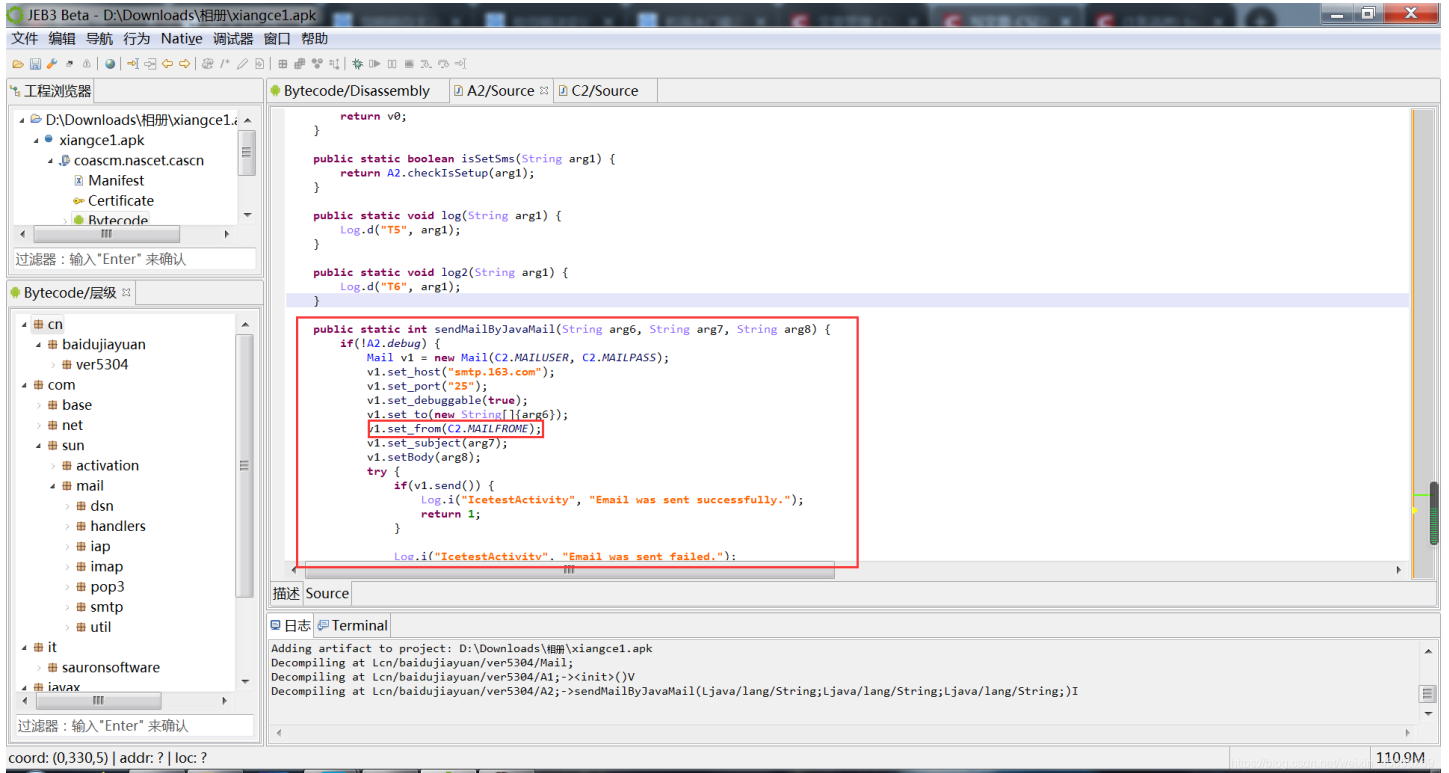
再交叉引用, 在fini段代码, 程序结束的时候会运行。

两个异或, 第一个异或是个“flag”反复异或得到一串字符串, 第二个异或是与干刚才得到的字符串的前4个再反复异或, 得到flag。

```
tmp=[0x40,0x35,0x20,0x56,0x5d,0x18,0x22,0x45,0x17,0x2f,
0x24,0x6e,0x62,0x3c,0x27,0x54,0x48,0x6c,0x24,0x6e,
0x72,0x3c,0x32,0x45,0x5b]
secret='flag'
v4=[0 for i in range(25)]
flag=[0 for i in range(25)]
for i in range(25):
    v4[i]=tmp[i]^ord(secret[i%4])
print(v4)
for i in range(25):
    flag[i]=tmp[i]^v4[i%4]
print(''.join(map(chr,flag)))
#flag{Act1ve_Defen5e_Test}
```

4.相册

apk文件, jeb打开, 题目提示邮件地址, ctrl+f搜索mail。找到关键函数sendMailByJavaMail。



c2变量是关键, 邮件来自那里就是邮件地址。



MAILFROM是加载外部so文件中NativeMethod.m()函数所返回的值。IDA打开apk解压的so文件, MTgyMTg0NjUxMjVAMTYzLmNvbQ==, base64解密就是flag。

5.[ACTF新生赛2020]easyre

UPX的壳，脱壳，拖入ida，

```
27 int i; // [esp+3Ch] [ebp-4h]
28
29 __main();
30 v4 = 42;
31 v5 = 70;
32 v6 = 39;
33 v7 = 34;
34 v8 = 78;
35 v9 = 44;
36 v10 = 34;
37 v11 = 40;
38 v12 = 73;
39 v13 = 63;
40 v14 = 43;
41 v15 = 64;
42 printf("Please input:");
43 scanf("%s", &v19);
44 if ( v19 != 'A' || v20 != 'C' || v21 != 'T' || v22 != 'F' || v23 != '{' || v27 != '}' )
45     return 0;
46 v16 = v24;
47 v17 = v25;
48 v18 = v26;
49 for ( i = 0; i <= 11; ++i )
50 {
51     if ( *(&v4 + i) != _data_start__[*((char *)&v16 + i) - 1] )
52         return 0;
53 }
54 printf("You are correct!");
55 return 0;
56 }
```

000007B1 _main:49 (4013B1) | https://blog.csdn.net/weixin_45055269

虽然输入是v19，但下面的v16-1作为下标去取某一数组的值，这些值已知，所以v16可求。试求一下v16，求出来提交竟然对了

```
v4=[42,70,39,34,78,44,34,40,73,63,43,64]
tmp='~}|{zyxwvutsrqponmlkjihgfedcba`_^]\ZYXWVUTSRQPONMLKJIHGFCDBA@?>=<;:9876543210/.-,+*)('+chr(0x27)+'&$$# !'
pos=[0 for i in range(12)]
for i in range(12):
    pos[i]=tmp.find(chr(v4[i]))+1
print(''.join(map(chr,pos)))
#U9X_1S_W6@T?
```

加上flag，提交就可

6.[SUCTF2019]SignIn

IDA打开

```
1 __int64 __fastcall main(__int64 a1, char **a2, char **a3)
2 {
3     char v4; // [rsp+0h] [rbp-4A0h]
4     char v5; // [rsp+10h] [rbp-490h]
5     char v6; // [rsp+20h] [rbp-480h]
6     char v7; // [rsp+30h] [rbp-470h]
7     char v8; // [rsp+40h] [rbp-460h]
8     char v9; // [rsp+B0h] [rbp-3F0h]
9     unsigned __int64 v10; // [rsp+498h] [rbp-8h]
10
11     v10 = __readfsqword(0x28u);
12     puts("[sign in]");
13     printf("[input your flag]: ", a2);
14     __isoc99_scanf("%99s", &v8); // v8输入
15     sub_96A(&v8, (__int64)&v9);
16     __gmpz_init_set_str((__int64)&v7, (__int64)"ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35", 16LL);
17     __gmpz_init_set_str((__int64)&v6, (__int64)&v9, 16LL); // 声明v6变量
18     __gmpz_init_set_str(
19         (__int64)&v4,
20         (__int64)"103461035900816914121390101299049044413950405173712170434161686539878160984549",
21         10LL);
22     __gmpz_init_set_str((__int64)&v5, (__int64)"65537", 10LL);
23     __gmpz_powm((__int64)&v6, (__int64)&v6, (__int64)&v5, (__int64)&v4); // rsa加密
24     if ( (unsigned int) __gmpz_cmp(&v6, &v7) )
25         puts("GG!");
26     else
27         puts("TTTTTTTTTTq1!");
28     return 0LL;
29 }
```

https://blog.csdn.net/weixin_45055269

sub_96A

```
1 size_t __fastcall sub_96A(const char *a1, __int64 a2)
2 {
3     size_t result; // rax
4     int v3; // [rsp+18h] [rbp-18h]
5     int i; // [rsp+1Ch] [rbp-14h]
6
7     v3 = 0;
8     for ( i = 0; ; i += 2 )
9     {
10         result = strlen(a1);
11         if ( v3 >= result )
12             break;
13         *(_BYTE *) (a2 + i) = byte_202010[(char)(a1[v3] >> 4)]; // 高位
14         *(_BYTE *) (a2 + i + 1LL) = byte_202010[a1[v3++] & 0xF]; // 低位
15     }
16     return result;
17 }
```

https://blog.csdn.net/weixin_45055269

将输入字符的ascii值（以十六位表示），转成字符格式即54—>'54'

有点意思，注意体会。

下面就是一个已知公钥的RSA算法，上脚本、


```

import gmpy2
import rsa
e=65537
n=103461035900816914121390101299049044413950405173712170434161686539878160984549
p=366669102002966856876605669837014229419
q=282164587459512124844245113950593348271
phin=(q-1)*(p-1)
d=gmpy2.invert(e,phin)
print(d)
enstr=0xad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35
destr=gmpy2.powmod(enstr,d,n)
flag0=hex(destr)[2:]
flag1=bytes.fromhex(flag0)
flag=str(flag1,'utf-8')
print(flag)

```

解出来的原文10进制表示，再转成16进制。hex与fromhex见本菜鸡曾写的CTF逆向中的字符与数字。

7.[GUET-CTF2019]re

UPX的壳，elf文件，所以在linux系统下脱壳，脱壳打开后

```

function Instruction Data Unexplored External symbol
IDA View-A Pseudocode-A Strings window Hex View-1 Structures Enums Im
1 BOOL8 __fastcall sub_4009AE(char *a1)
2 {
3     if ( 1629056 * *a1 != 166163712 )
4         return 0LL;
5     if ( 6771600 * a1[1] != 731332800 )
6         return 0LL;
7     if ( 3682944 * a1[2] != 357245568 )
8         return 0LL;
9     if ( 10431000 * a1[3] != 1074393000 )
10        return 0LL;
11    if ( 3977328 * a1[4] != 489211344 )
12        return 0LL;
13    if ( 5138336 * a1[5] != 518971936 )
14        return 0LL;
15    if ( 7532250 * a1[7] != 406741500 )
16        return 0LL;
17    if ( 5551632 * a1[8] != 294236496 )
18        return 0LL;
19    if ( 3409728 * a1[9] != 177305856 )
20        return 0LL;
21    if ( 13013670 * a1[10] != 650683500 )
22        return 0LL;
23    if ( 6088797 * a1[11] != 298351053 )
24        return 0LL;
25    if ( 7884663 * a1[12] != 386348487 )
26        return 0LL;
27    if ( 8944053 * a1[13] != 438258597 )
28        return 0LL;
29    if ( 5198490 * a1[14] != 249527520 )
30        return 0LL;
000009AE sub_4009AE:1 (4009AE)
https://blog.csdn.net/Welxin_45055269

```

直接找到关键函数，简单的除法就可逆向，考虑z3求解，由于都是整数，声明变量时int即可

```

from z3 import *
s=Solver()
a1=[0 for i in range(32)]
for i in range(32):
    a1[i]=Int('a1['+str(i)+''])
s.add ( 1629056 * a1[0] == 166163712 )
s.add ( 6771600 * a1[1] == 731332800 )
s.add ( 3682944 * a1[2] == 357245568 )
s.add ( 10431000 * a1[3] == 1074393000 )
s.add ( 3977328 * a1[4] == 489211344 )
s.add ( 5138336 * a1[5] == 518971936 )
s.add ( 7532250 * a1[7] == 406741500 )
s.add ( 5551632 * a1[8] == 294236496 )
s.add ( 3409728 * a1[9] == 177305856 )
s.add ( 13013670 * a1[10] == 650683500 )
s.add ( 6088797 * a1[11] == 298351053 )
s.add ( 7884663 * a1[12] == 386348487 )
s.add ( 8944053 * a1[13] == 438258597 )
s.add ( 5198490 * a1[14] == 249527520 )
s.add ( 4544518 * a1[15] == 445362764 )
s.add ( 3645600 * a1[17] == 174988800 )
s.add ( 10115280 * a1[16] == 981182160 )
s.add ( 9667504 * a1[18] == 493042704 )
s.add ( 5364450 * a1[19] == 257493600 )
s.add ( 13464540 * a1[20] == 767478780 )
s.add ( 5488432 * a1[21] == 312840624 )
s.add ( 14479500 * a1[22] == 1404511500 )
s.add ( 6451830 * a1[23] == 316139670 )
s.add ( 6252576 * a1[24] == 619005024 )
s.add ( 7763364 * a1[25] == 372641472 )
s.add ( 7327320 * a1[26] == 373693320 )
s.add ( 8741520 * a1[27] == 498266640 )
s.add ( 8871876 * a1[28] == 452465676 )
s.add ( 4086720 * a1[29] == 208422720 )
s.add ( 9374400 * a1[30] == 515592000 )
s.add(5759124 * a1[31] == 719890500)
print(s.check())
answer=s.model()
print(answer)

```

算出来，32位，又仔细去看题目，发现a1[6]出题人就没给，所以开始试，是1。

本题题目的名字是re，所以我将相关py文件都命名成re.py，这就出现了问题，linux和win下都会自动将我写的这个re文件作为配置文件，导致报错。

z3跑出来一长溜数字，自然高兴，大致一看是按从大到小顺序排列，实则不然，16和17两个位置颠倒，细节注意。

```

a1 = [0]*32
a1[31] = 125
a1[30] = 55
a1[29] = 51
a1[28] = 51
a1[27] = 57
a1[26] = 51
a1[25] = 48
a1[24] = 99
a1[23] = 49
a1[22] = 97
a1[21] = 57
a1[20] = 57
a1[19] = 48
a1[18] = 51
a1[16] = 97
a1[17] = 48
a1[15] = 98
a1[14] = 48
a1[13] = 49
a1[12] = 49
a1[11] = 49
a1[10] = 50
a1[9] = 52
a1[8] = 53
a1[7] = 54
a1[5] = 101
a1[4] = 123
a1[3] = 103
a1[2] = 97
a1[1] = 108
a1[0] = 102
for i in range(32):
    if i == 6:
        continue
    print(chr(a1[i]), end="")

```

8.[ACTF新生赛2020]rome

```

v4 = v13;
for ( i = 0; i <= 15; ++i )
{
    if ( *((_BYTE *)&v1 + i) > '@' && *((_BYTE *)&v1 + i) <= 'Z' )
        *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
    if ( *((_BYTE *)&v1 + i) > '`' && *((_BYTE *)&v1 + i) <= 'z' )
        *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
}
for ( i = 0; i <= 15; ++i )
{
    result = (unsigned __int8)*(&v15 + i);
    if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
        return result;
}
result = printf("You are correct!");
}

```

https://blog.csdn.net/weixin_45055269

ida打开，逻辑清晰，爆破处理

```

x = [81,115,119,51,115,106,95,108,122,52,95,85,106,119,64,108]
flag = ''
for k in range(0,16):
    for i in range(0,127):
        z=i
        if i > 64 and i <= 90:
            i = (i-51)%26 + 65
        if i > 96 and i <= 122:
            i = (i-79)%26 + 97
        if(i == x[k]):
            flag += chr(z)

print(flag)
#Cae3ar_th4_Gre@t

```

对爆破脚本的新理解，z=i，保存原始状态，flag+=chr(z)，将原始状态加入flag。（之前一直不理解为何z=i再赋值一遍，笑哭.png）

9.[FlareOn4]login

给了一个网页，ctrl+u看源码，找到核心部分

```

document.getElementById("prompt").onclick = function () {
    var flag = document.getElementById("flag").value;
    var rotFlag = flag.replace(/[a-zA-Z]/g, function(c){return String.fromCharCode((c <= "Z"
    ? 90 : 122) >= (c = c.charCodeAt(0) + 13) ? c : c - 26)});
    alert(rotFlag)
    if ("PyvragFvqrYbtvafNerRnfl@syner-ba.pbz" == rotFlag) {
        alert("Correct flag!");
    }
}

```

https://blog.csdn.net/weixin_45055269

其中涉及的js函数均可百度，想不明白就在纸上算算。**ROT13替换**，为啥是13，而不是别的，因为26个英文字母，一半是13，正着走或倒着走均可走到13位的替换位置，这样想可以出一个1到10的，整个ROT5。具体做法：把密文在输入一遍，在if前加一个 alert(rotFlag)，点击就会弹出flag

10.[V&N2020 公开赛]strangeCpp

是c++程序，搜索字符串找到的内容一脸懵，

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

根据提示，找到字符串，发现一串可疑字符串被sub_140013580引用，看着像是flag，搜索引用找到代码，分析逻辑发现是，找一个数经过sub_140011384运算得607052314，且这个数小于等于14549743，这个值可以爆破出来，爆破出来的结果和byte_140021008异或。

整数爆破

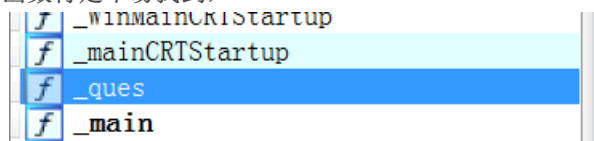
```

v8=0
for i in range(14549743):
    tmp=((i<<8)^(i>>12))*291&0xffffffff
    if tmp==607052314:
        v8=i
        break
a=[0x26, 0x2C, 0x21, 0x27, 0x3B, 0x0D, 0x04, 0x75, 0x68, 0x34,
    0x28, 0x25, 0x0E, 0x35, 0x2D, 0x69, 0x3D, 0x6F, 0x6D, 0x00]
print(v8)
for j in range(17):
    print(chr((v8^a[j])&0xff),end="")
#123456
#fLag{MD5(theNum)}

```

11. [BJDCTF2020]easy

Do you find me?一看就知道，真正的函数肯定不易找到，



ques函数，找到了，这个函数提到了几个概念LODWORD, HIWORD, SHIDWORD。我没找到具体的这几个的定义，只知道这是IDA的宏定义，参考了师傅的WP，找到了这个（下图）：

```
#define LOWORD(l) ((WORD)((DWORD_PTR)(l) & 0xffff))
```

```
#define HIWORD(l) ((WORD)((DWORD_PTR)(l) >> 16))
```

这是windef.h头文件中对宏LOWORD和HIWORD的定义。

作用分别是取出无符号长整型参数的高16位和低16位。

因为一个长整型占32位，其中高低16位的值可能有不同的意义，需要通过这2个宏分别取出来使用。取出来的结果是一个无符号短整型的值。

其原理正如定义那样，取低16位的宏LOWORD使用按位与操作符与数字0xffff运算，而数字0xffff是一个低16位全为1的数字，那么对其位与操作可以得到参数的低16位。

而取高16位的宏HIWORD则更简单，只需将参数右移16位，剩下的就是原高16位的值了。

类比过来就可，V14的值就是64位，SHIDWORD这个的S是signed有符号的意思，然后看向V2赋值的算法。看着花哨，实际就是v14。

```

{
    v2[v16++] = (((SHIDWORD(v14) >> 31) ^ (((SHIDWORD(v14) >> 31) ^ v14) - (SHIDWORD(v14) >> 31)) & 1)
                - (SHIDWORD(v14) >> 31));
    v14 /= 2i64;
}

```

SHIDWORD本身是v14的高32位，由于是有符号数，最高位为0，右移31位刚好是0。

接下来，打印'*'字符，形成flag

```

#include<iostream>
using namespace std;
int main() {
    char v2[10][100] = {
        "10001001001111110001111111111110100111111000110001",
        "10001010101000010010001000010010100100001000111001",
        "11111111111000011100001000010011111111001000110101",
        "10001100011000010110001000010000100100001000110011",
        "10001100011111110001111110010000100100001111110001",
    };
    int v15 = 0;
    for (int i = 0; i <= 4; i++) {
        for (int j = 0; j < 50; ++j) {
            if (v2[i][j]=='1')
            {
                cout << "*";
                ++v15;
            }
            else {
                cout << " ";
                ++v15;
            }
        }
        if (v15%5==0)
        {
            cout << " ";
        }
    }
    cout << endl;
}
}

```

```
× × × ××××× × × ××××× ××××× × × ××××× × × × × ×
× × × × × × × × × × × × × × × × × × × ×
×××××× ×××××× × ×××× × × ××××× ×××× × × × × × ×
× × × × × × × ×× × × × × × × × × × × × × × ×
× × × × ××××× × × ××××× × × × × ××××× × × ×
```

D:\proj\[BJDCTF2020]easy\Debug\[BJDCTF2020]easy.exe (进程 10440)已退出，返回代码为：0。
按任意键关闭此窗口...