

ISCC 2016 逆向部分 writeup

原创

GlodsNow 于 2016-05-25 13:23:56 发布 5051 收藏 1

分类专栏: [writeup](#) 文章标签: [ISCC2016 逆向](#) [GoldsNow](#) [Mirage](#) [writeup](#)

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/GlodsNow/article/details/51497679>

版权



[writeup](#) 专栏收录该内容

2 篇文章 0 订阅

订阅专栏

ISCC2016 逆向部分

by GoldsNow

做了这套题目感觉涨了不少的姿势。。渣渣就只能用渣渣的方法

[题目下载](#)

Help me

描述: I've got a difficult task and I can't solve it. I need your help!

思路: 爆破

ELF64位的程序, 直接在IDA中打开, 可以看出来思路应该还是比较清晰
先点进去main函数来进行分析。

```
23 puts("Hey man, could you help me decrypt some data?");
24 printf("Gimme the key sequence : ", 0LL);
25 while ( __isoc99_scanf("%lu", &u9) != -1 )
26     v10 += u9;
27 v10 = (unsigned __int16)v10;
28 v11 = (unsigned __int16)v10;
29 for ( i = 0; i <= 15; ++i )
30 {
31     if ( !(v11 & 1) )
32         ++v5;
33     v11 >>= 1;
34 }
35 if ( v5 != 3 )
36 {
37     puts("No!! This combination doesn't meet the requirements!");
38     exit(0);
39 }
40 puts("Ok, now let me check your combination...");
41 for ( j = 0; j <= 5; ++j )
42     word_602080[j] ^= v10;
43 MD5((int64)word_602080, 22LL, (int64)&s1);
44 v12 = word_602080;
45 for ( k = 0; k <= 21; ++k )
46     byte_6020C0[(signed int64)k] -= *((_BYTE *)v12 + k);
47 if ( !memcmp(&s1, &unk_6020A0, 16uLL) )
48 {
49     puts("Alright! It's correct!!");
```

```

for ( i = 0; i <= 15; ++i )
{
    if ( !(v11 & 1) )
        ++v5;
    v11 >>= 1;
}

```

先对这个一段代码进行分析，经过位运算，来判断二进制数字有多少个0
很明显 这里只需要三个0

这里的MD5是个比较坑的地方，C语言并没有自带的库，也不知道它到底进行的是什么。只知道他是与 unk_6020A0 来比较的
有尝试对这个地址的数据进行MD5的解密，可惜最终是失败的了的。

换一个思路，我们可以看到这里的 word_602080 经过了异或运算 然后 byte_6020C0 减去 这个的值然后输出最终的flag 所以可以从这个角度来解决题目，这里面值有 V10也就是输入的那一个数字是不知道的，可以通过暴力破解来实现。
我在做题目的时候用了C语言来判断，判断第五个字母是不是 '{'

附上我的c程序可以参考一下。

```

unsigned char a[]={0x34,0x12,0x78,0x56,0xBC,0x9A,0xFF,0xED,0xEF,0xBE,0x7F,0x22,0xC3,0x90,0x76,0x82,0xAD,0x99,0x2E,0x14,0x7C,0x80};
unsigned char b[]={0x22,0x3F,0xD8,0xEB,0xCC,0xD2,0x42,0x87,0x61,0x75,0x01,0x09,0x27,0xF9,0xDC,0xE8,0x16,0xFC,0x5F,0x89,0xB3,0xFD};
__int16 c[6]={0x3412,0x7856,0xbc9a,0xffed,0xefbe,0x7f22};
__int16 d[6];
__int16 x1=0,x2=0xffff,x=0;
for(int i=0;i<=15;i++) // 三重for循环将 v10的可能性都列出来了
    for(int j=i+1;j<=15;j++)
        for(int m=j+1;m<=15;m++)
        {
            x2==0xffff;
            x1((__int16)(pow(2,i)+pow(2,j)+pow(2,m)));
            x=x2-x1; //这三行是为了得到V10的可以理解一下

            for(int n=0;n<=5;n++)
            {
                d[n]=c[n]^x;
                a[2*n]=d[n]>>8;
                a[2*n+1]=(__int8) d[n];
            }
            if(b[4]-a[4]=='{') //如果是 '{'就输出 可能的flag
            {
                printf("%x ",x);
                for(int i=0;i<22;i++)
                    printf("%c",b[i]-a[i]);
                printf("\n");
            }
        }
}

```

破解加密软件

描述：我方截获了敌方的一款加密程序和一段密文，尝试通过对此程序进行分析实现对密文的解密，解密后明文32位小写MD5值作为flag提交。

思路：扣出程序中所包含的一个加密解密系统，得到密钥解密

吾爱破解对于这个题目也有比较详细的解题过程 [链接](#)

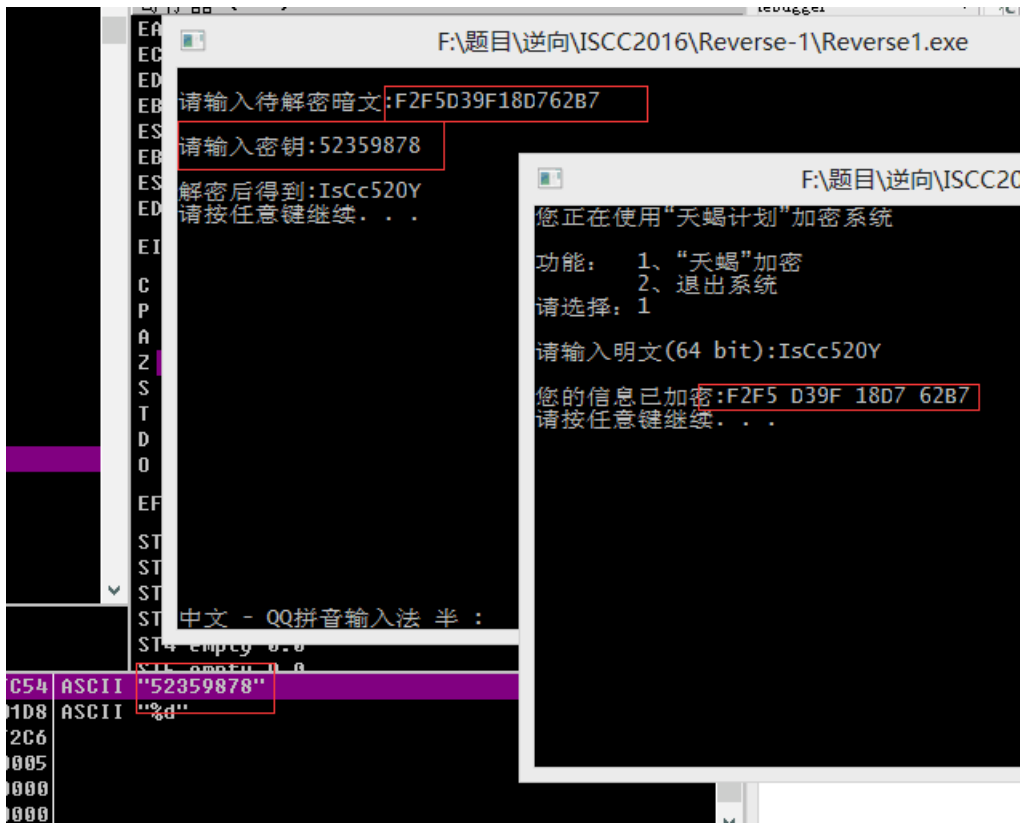
地址	反汇编	字符串内容
00EE1148	push Reverse.00EF0204	\n请输入明文 (64 bit):
00EE1180	push Reverse.00EF021C	\n明文过长, 请重新输入:
00EE1273	push Reverse.00EF01D8	%d
00EE12A5	push Reverse.00EF01DC	\n您的信息已加密:
00EE12BA	push Reverse.00EF01F0	%c
00EE12D0	push Reverse.00EF01F4	\n
00EE12E3	push Reverse.00EF01F8	\n
00EE12ED	push Reverse.00EF01FC	Pause
00EE144E	push Reverse.00EF0204	\n请输入明文 (64 bit):
00EE14A0	push Reverse.00EF021C	\n明文过长, 请重新输入:
00EE14CD	push Reverse.00EF0234	\n请输入密钥 (64 bit):
00EE1500	push Reverse.00EF024C	\n密钥长度错误, 请重新输入:
00EE1543	push Reverse.00EF01DC	\n您的信息已加密:
00EE1558	push Reverse.00EF01F0	%c
00EE156E	push Reverse.00EF01F4	\n
00EE1581	push Reverse.00EF01F8	\n
00EE158B	push Reverse.00EF01FC	Pause
00EE15CE	push Reverse.00EF0268	\n请输入待解密密文:
00EE1622	push Reverse.00EF027C	\n密文长度错误, 请重新输入:
00EE164D	push Reverse.00EF0298	\n请输入密钥:
00EE1680	push Reverse.00EF024C	\n密钥长度错误, 请重新输入:
00EE16C3	push Reverse.00EF02A8	\n解密后得到:
00EE16D8	push Reverse.00EF01F0	%c
00EE16EB	push Reverse.00EF01F8	\n
00EE16F5	push Reverse.00EF02B8	pause
00EE1987	mov ecx, Reverse.00EF02C0	iscc201b
00EE19B9	push Reverse.00EF02CC	cls
00EE19C6	push Reverse.00EF0320	您正在使用加密解密系统\n
00EE19D0	push Reverse.00EF0338	\n功能: \t1、使用自定义密钥加密\n\t2、使用自定义密钥解密\n\t3、退出系统\n请选择:
00EE1A8E	push Reverse.00EF02CC	%d
00EE1A1B	push Reverse.00EF02D0	您正在使用“天蝎计划”加密系统\n
00EE1A25	push Reverse.00EF02F0	\n功能: \t1、“天蝎”加密\n\t2、退出系统\n请选择:
00EE1FC1	push Reverse.00EF0384	mscree.dll
00EE1FD2	push Reverse.00EF039C	CorExitProcess
00EE204B	push Reverse.00EF4000	t马
00EE22D3	push Reverse.00EF03AC	COMSPEC
00EE2319	mov [local.4], Reverse.00EF03B4	/c
00EE2374	mov ecx, Reverse.00EF03B8	cmd.exe
00EE274C	mov eax, dword ptr ds:[0xEE003C]	%

原题可以看到的程序

通过OD载入然后搜索里面的字符串可以发现 程序有两个功能，一个就是原题所给的 输入字符串就加密，其实里面还有一个程序就是 通过给与密钥就能够加密与解密

通过修改跳转，或者修改 call 的内容等等其他 方法能够让程序运行到那一段 这样就好办了。但是 还不知道密钥是什么。想法是密钥肯定会在 初始的加密程序中引用

对程序进行单步跟踪。一步一步运行 到达一定地方的时候堆栈中就出现了如下可惜的八位数字，并且这个不会随着你的输入而改变数字，猜测其为密钥



果然没错 一下子就出来了

这里有一个地方就是 直接用OD跑出来的值会有一个空格多出来也不知道是因为什么。

菜鸟的逆袭

描述：“小明是个游戏新手，一天他闲得无聊去找基友打L4D2，基友嫌他太菜，不愿带他，无奈小明百般纠缠，便给了他一个程序，并且告诉他：“如果你能得到正确答案，我就把我毕生混野绝学传授给你。”聪明的你能帮助小明得到正确的答案吗？

测试环境：干净的WinXP SP3”

找不到动态调试的软件。只能够静态分析，用IDA打开。

```

v8 = *(v11 + 12);
v9 = &Irp->AssociatedIrp.MasterIrp->Type;
v10 = *(v11 + 8);
if ( v8 == 2236420 )
{
    v3 = dword_10F80;
    v4 = dword_10F84;
    v5 = dword_10F88;
    v6 = dword_10F8C;
    v7 = dword_10F90;
    for ( i = 0; i < 5; ++i )
        *(&v3 + i) ^= *v9;
    v14 = sub_10950(&v3, 20);
    if ( v14 == -1276902268 )
    {
        dword_10F80 = v3;
        dword_10F84 = v4;
        dword_10F88 = v5;
        dword_10F8C = v6;
        dword_10F90 = v7;
        KeSetEvent(&Object, 0, 0);
    }
    else
    {
        DbgPrint("Hint:This value is a widely used hex value in computer science.\n");
        DbgPrint("it consists of two words of 4 characters with hex letters of 'A' to 'F'(inclusive).\n");
        DbgPrint("The first word has the same meaning as \"deceased\"\n");
        DbgPrint("The second word represents a kind of meat.\n");
    }
}

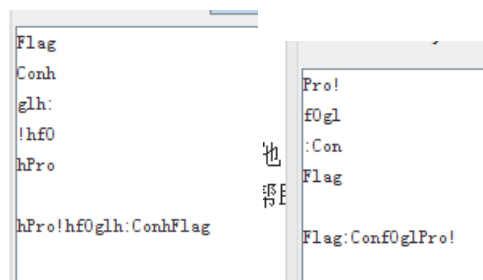
```

可以通过提示 推断出来 这里有两个词，分别是DEAD和BEEF

正好属于A-F 因为是A-F 属于0-F的范围 都符合16进制的值。所以上面的V9就是0xdeadbeef或者说0xDEADBEEF

通过将dword_10F80-dword_10F90与V9异或输出，能够得到一串看起来很像flag的值但是 要靠程序接下来分析 将字符串进行了转化

orPh0fh!:lhghnoCgalF



这样转化一下就出来了，， 是不是很简单，， ， 但是要去想到却很难。。

矛盾

描述: To be or not to be, that is the question.

思路 去除TLS，暴力

参考资料 <http://www.2cto.com/Article/201303/197705.html>

IDA载入

```
sub_43B7E0 0043BA10 CC 89 6C 82 4C 00 E8 1B C7 FF FF 68 8B B9 4A 00
sub_43B840 0043BA20 E8 85 CB FF FF 59 C3 CC CC CC CC CC CC CC CC
sub_43B8A0 0043BA30 55 8B EC 81 EC 18 01 00 00 53 56 57 8D BD E8 FE
sub_43B900 0043BA40 FF FF B9 46 00 00 00 B8 CC CC CC CC F3 AB 83 7D
sub_43B960 0043BA50 0C 01 75 1E C6 45 AC 00 6A 4F 6A 00 8D 45 AD 50
sub_43B9C0 0043BA60 E8 9A CB FF FF 83 C4 0C 6A 00 E8 2E DD FF FF 83
sub_43BA11 0043BA70 C4 04 52 8B CD 50 8D 15 9C BA 43 00 E8 2D CC FF
sub_43BA11 0043BA80 FF 58 5A 5F 5E 5B 81 C4 18 01 00 00 3B EC E8 F3
sub_43BA11 0043BA90 D5 FF FF 8B E5 5D C2 0C 00 8D 49 00 01 00 00 00
sub_43BA11 0043BAA0 A4 BA 43 00 AC FF FF FF 50 00 00 00 B0 BA 43 00
sub_43BB90 0043BAB0 73 7A 4D 72 67 00 CC CC CC CC CC CC CC CC CC CC
sub_43BC10 0043BAC0 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
sub_43BC80 0043BAD0 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
sub_43BCF0 0043BAE0 55 8B EC 81 EC 18 01 00 00 53 56 57 8D BD E8 FE
sub_43BD40 0043BAF0 FF FF B9 46 00 00 00 B8 CC CC CC CC F3 AB 83 7D
sub_43BDC0 0043BB00 0C 01 75 23 C6 45 AC 00 6A 4F 6A 00 8D 45 AD 50
sub_43BE40 0043BB10 E8 EA CA FF FF 83 C4 0C 6A 01 E8 7E DC FF FF 83
sub_43BEA0 0043BB20 C4 04 E8 B8 CC FF FF 52 8B CD 50 8D 15 50 BB 43
sub_43BF00 0043BB30 00 E8 78 CB FF FF 58 5A 5F 5E 5B 81 C4 18 01 00
sub_43BF60 0043BB40 00 3B EC E8 3E D5 FF FF 8B E5 5D C2 0C 00 8B FF
sub_43BFC0 0043BB50 01 00 00 00 58 BB 43 00 AC FF FF FF 50 00 00 00
sub_43C020 0043BB60 64 BB 43 00 73 7A 4D 72 67 00 CC CC CC CC CC CC
sub_43C080 0043BB70 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
sub_43C0E0 0043BB80 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
sub_43C140 0043BB90 FF 00 50 04 50 00 00 00 00 50 57 57 00 00 10 FF
```

找到TLScallback 在010中打开它用00填充此部分 保存文件就能够进入程序的主函数部分了，但是还是不能够得到flag 有点小郁闷

通过查看字符串来寻找有没有弹出窗口，运气不错发现一个弹出窗口。并且在上面也出现了flag的字样。

```
3 v31 |= 0x69u;
4 v32 |= 0x61u;
5 v33 |= 0x6Eu;
6 v8 = 'f';
7 v9 = 'l';
8 v10 = 'a';
9 v11 = 'g';
10 v12 = ':';
11 v13 = '{';
12 v0 = '}' ;
13 v15 = '}' ;
14 for ( i = 0; i < 0x1E; ++i )
15 {
16     v0 = i;
17     v14[i] |= *(&v17 + i);
18 }
19 if ( dword_4C8164 == 1 )
20 {
21     for ( j = 0; j < v34; ++j )
22         *(&Text + j) = *(&v8 + j);
23     MessageBoxW(0, &Text, 0, 0);
24     ((void (*)(void))sub_439086)();
25 }
26 v1 = v0;
27 sub_4386AE(&savedregs, &dword_43E320);
28 return sub_439086((unsigned int)&savedregs ^ v35, v1);
29 }
```

函数的上有几个 sub_4385FF 推测他用的就是memset 将目标的地址段 值置零。所以这里就会使最终的flag 最后处理 的是直接打开OD跳转到相关的语段 跑一下就出来了。

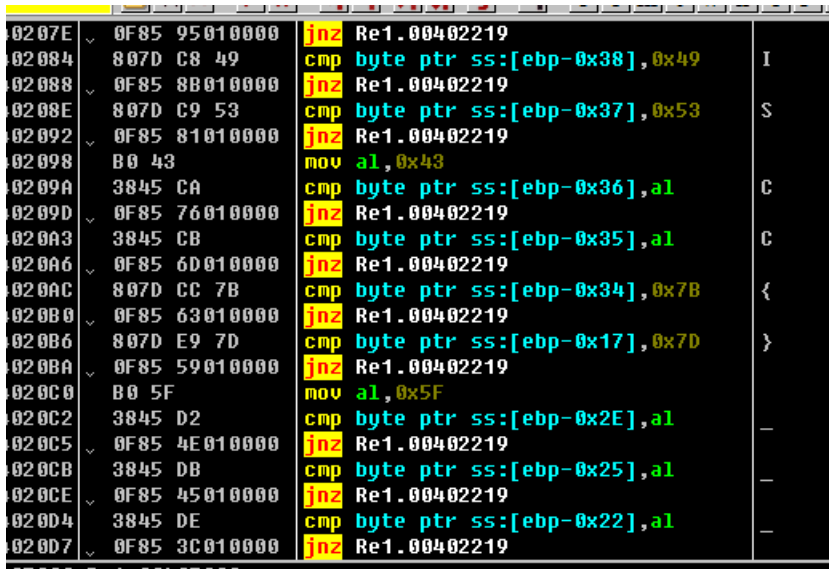
Anti

描述: Every technique has its anti.

这个题目就是反调试，绕过dll文件中的对调试器和模拟器的检测。。

操作方法，具体就不介绍了，就是在OD中一步一步，对于有些 call 直接就nop掉一些明显要推出的函数就强制跳转或者也nop了

进入主程序之后，难度就会明显下降



```
0207E 0F85 95010000 jnz Re1.00402219
02084 807D C8 49 cmp byte ptr ss:[ebp-0x38],0x49 I
02088 0F85 88010000 jnz Re1.00402219
0208E 807D C9 53 cmp byte ptr ss:[ebp-0x37],0x53 S
02092 0F85 81010000 jnz Re1.00402219
02098 B0 43 mov al,0x43
0209A 3845 CA cmp byte ptr ss:[ebp-0x36],al C
0209D 0F85 76010000 jnz Re1.00402219
020A3 3845 CB cmp byte ptr ss:[ebp-0x35],al C
020A6 0F85 6D010000 jnz Re1.00402219
020AC 807D CC 7B cmp byte ptr ss:[ebp-0x34],0x7B {
020B0 0F85 63010000 jnz Re1.00402219
020B6 807D E9 7D cmp byte ptr ss:[ebp-0x17],0x7D }
020BA 0F85 59010000 jnz Re1.00402219
020C0 B0 5F mov al,0x5F
020C2 3845 D2 cmp byte ptr ss:[ebp-0x2E],al -
020C5 0F85 4E010000 jnz Re1.00402219
020CB 3845 DB cmp byte ptr ss:[ebp-0x25],al -
020CE 0F85 45010000 jnz Re1.00402219
020D4 3845 DE cmp byte ptr ss:[ebp-0x22],al -
020D7 0F85 3C010000 jnz Re1.00402219
```

很容易就能够找到主要的那个对比的地址，最后有一段看起来很复杂的加密方法，，，分析半天没分析出来，后来一看就32位的，，就想到了可能是用了MD5加密方法，，最终在XMD5一下子就出来flag了。

GoGoGo

描述：Mission is a go.

这个题目用了GO语言。问了很多人都说是用动态分析做出来了，但是我最后是通过静态分析，很容易就做出来了。

IDA的分析是非常规的。只能靠猜

```
v38 = "Please input : ";
v39 = off_4DC4C0[1];
v40 = 0LL;
v41 = 0LL;
if ( &v30 == -88 )
    LODWORD(v40) = 0;
v42 = &v40;
v43 = 1;
v44 = 1;
v30 = &unk_4A10A0;
v31 = &v38;
sub_41FA10(&v42, a2, a3, v3);
v4 = v42;
v5 = v42;
*v42 = v32;
++v5;
*v5 = v33;
++v5;
v30 = v4;
v31 = v43;
v32 = v44;
sub_42C510(v5, &v34, v6, v7, v8, v9);
sub_400C00(v5, &v34);
```

```
v36 = v30;
v37 = v31;
v32 = &unk_4D73F0;
v33 = *(&off_4D73E0 + 1);
v10 = &off_4D73E0 + 2;
sub_446A70(&v34, (&off_4D73E0 + 2), v11, v30, v12, v13);
v16 = v34;
v36 = v34;
v37 = v35;
if ( v35 != 38 || (v30 = v34, v31 = 38, sub_400CB0(&v34, v10, v14, v34, v15), v32 != 1) )
{
    v38 = "Wrong!!!";
    v39 = off_4DCCE0[1];
    v40 = 0LL;
    v41 = 0LL;
    if ( &v30 == -88 )
        LODWORD(v40) = 0;
    v42 = &v40;
    v43 = 1;
    v44 = 1;
    v30 = &unk_4A10A0;
    v31 = &v38;
    sub_41FA10(&v42, v10, v14, v16);
    v24 = v42;
    v25 = v42;
    *v42 = v32;
    ++v25;
    *v25 = v33;
    v30 = v24;
    v31 = v43;
    v32 = v44;
    result = sub_42C510((v25 + 1), &v34, v26, v27, v28, v29);
}
else
{
    v38 = "Congratz!!";
    v39 = off_4DB7A0[1];
    v40 = 0LL;
    v41 = 0LL;
    if ( &v30 == -88 )
        LODWORD(v40) = 0;
    v42 = &v40;
    v43 = 1;
    v44 = 1;
    v30 = &unk_4A10A0;
    v31 = &v38;
    sub_41FA10(&v42, v10, v14, v16);
    v17 = v42;
    v18 = v42;
    *v42 = v32;
    ++v18;
    *v18 = v33;
    v30 = v17;
    v31 = v43;
    v32 = v44;
    result = sub_42C510((v18 + 1), &v34, v19, v20, v21, v22);
}
}
```


截图太累 直接把代码复制了，， 通过对比 可以看出 这个 'sub_42C510' 才是输出的结束语句
input 那边还需要一个读取字符的语句 就猜测是 'sub_446A70' 这语段是输出语段。

主要起作用的当然是 if语句里的那个 sub_400CB0 了

```
6 | {
7 |   if ( v17 >= v16 )
8 |   {
9 |     sub_40F350(v6, v7, v16, v14);
10 |    BUG();
11 |   }
12 |   v7 = (16LL * v6 | (*(v15 + v17) >> 4));
13 |   {
14 |     sub_40F350(v6, v7, v16, v14);
15 |    BUG();
16 |   }
17 |   *(v15 + v17++) = v7;
18 | }
19 | while ( v16 > v17 );
20 | }
21 | if ( &v23 == -84 )
22 |   *&v29 = &v29;
23 | LODWORD(v18) = (loc_42806C)(&v29, unk_506180);
24 | *&v29 = unk_506180;
25 | if ( !v18 )
26 |   v0 = 0;
27 | v21 = v18;
28 | result = 0LL;
29 | if ( v19 > 0 )
30 | ,
```

这里有一个 字符转化和一个内存中的数据很可疑，， 刚开始以为是输入的语句经过处理变成这样， 但是实际上我错了。是这个内存中的 数据 经过这个处理会出现一段base64加密过的数据。。

附上代码

```
int b[52]={0xA5,0xD6,0x87,0x86,0xA5,0x33,0x37,0x03,0xE4,0x75,0xE4,0xB6,0x95,0xA7,0xC6,0xD6,0xE4,0x44,0x94,0x53,0xE4,0xA6,0xB6,0x53,0xD4,0x23,0xD4,0x77,0xD4,0xA7,0x46,0xB6,0xA5,0x74,0x55,0x13,0xD4,0xA6,0x36,0x87,0x95,0x75,0x55,0x43,0x95,0xA6,0x15,0x03,0x95,0x33,0x03,0xD3};
for(int j=0;j<52;j++)
{
    printf("%c",(16*b[j]|b[j]>>4));
}
```

就是这么简单，， 然后将这个进过base64解密一下就出来了 我也觉的很意外。